Dynamic position calibration

M. de Jong

January 21, 2022

Abstract

The dynamic position calibration of the detector using the Jpp software framework is presented. The procedure is based on a global fit of a model to the acoustic data.

1 Introduction

The KM3NeT research infrastructure comprises several large 3D-arrays of so-called optical modules deployed in the deep waters of the Mediterranean Sea. These optical modules consist of a glass sphere which houses the photo-multiplier tubes (PMTs) and the readout electronics that are used for the detection of neutrinos. The optical modules are arranged on strings which are anchored on the seabed and held vertically by the buoyancy of the glass spheres, further supported by a buoy on top. To achieve the envisaged angular resolution of the detected neutrinos, the positions of the optical modules should be measured with an accuracy of about $20\ \mathrm{cm}$. Due to varying sea currents, the strings sway in the sea water. To accurately follow the movements of the optical modules, their positions should be measured every 10 minutes or so. The repeated measurements of the positions of the optical modules are referred to as "dynamic position calibration".

For the dynamic position calibration, a system of acoustic emitters and acoustic receivers is deployed. The acoustic emitters are mounted on so-called tripods which are positioned on the seabed outside the detector footprint. They are battery powered and autonomously "ping" at regular time intervals. Two kinds of acoustic receivers are mounted on the detector. A piezo sensor is glued on the inside of each glass sphere and a hydrophone is mounted on some of the anchors. The hydrophone is readout by a (base) module which is also mounted on the anchor.

To limit the power consumption of an acoustic emitter, a typical sequence consisting of 11 pings with a 5 seconds interval is repeated every 10 minutes. The signal of a given emitter has a characteristic frequency and amplitude modulation. A signal is detected by matching a corresponding waveform to the raw data from an acoustic receiver. For every matched signal, the time-of-arrival is recorded on shore. To distinguish the different signals, each waveform has a unique identifier which is also recorded. The recorded data are referred to as "acoustic data". Due to some features in the hardware/firmware of the different acoustic receivers, more than one waveform is used to optimally detect the signal from a specific emitter.

The acoustic data are continually taken during normal operation of the detector and stored in the central database in table "toashort". The list of columns of this table includes:

name	data type
DETID	std::string
RUN	int
RUNNUMBER	int
UNIXTIMEBASE	double
DOMID	int
EMITTERID	int
TOA_S	double
QUALITYFACTOR	int

In this, DETID corresponds to the identifier of the detector (OID in table "detectors"), RUN to the number of the data taking run, DOMID to the identifier of the module which relates to the acoustic receiver, and EMITTERID to the identifier of the expected waveform that is used to detect a signal from a specific acoustic emitter. The measured time-of-arrival and the quality of the measurement are stored in columns TOA_S and QUALITYFACTOR, respectively.

The default geometry of the detector can be used to relate the identifier of a module to its string and floor number in the detector. The UTC time-of-arrival (i.e. time corrected for the phase of the clock of the readout) at string i and floor j can be obtained as follows.

$$t_{ij} \equiv \texttt{UNIXTIMEBASE} + \texttt{TOA\_S} \tag{1}$$

This time can be compared to the UTC time of the PMT data (see JDAQUTCExtended). By correlation of the UTC times, the dynamic position calibration can consistently be applied to the PMT data that are used to reconstruct the neutrino interactions in the detector.

In the following, the dynamic position calibration of the detector using the Jpp software framework is presented (see https://git.km3net.de/common/jpp). The corresponding interfaces, classes and methods are organised in name space JACOUSTICS. Unless otherwise stated, the units for the length and time are m and s, respectively.

2 Sound velocity

The velocity of sound in the deep-sea water roughly amounts to $1550\ \mathrm{m/s}$. The actual value depends on the site and the depth (and may vary slowly over the year). For the actual range of depths, the depth dependence is –to good approximation– linear. The assumed sound velocity and depth dependence is defined in JSoundVelocity. This class provides for an implementation of the methods:

```
double getDistance(const double t_s,
                   const double z1,
                   const double z2);

double getTime(const double D_m,
               const double z1,
               const double z2);
```

which return the distance and propagation time for a given propagation time and distance, respectively. The depth dependence is handled via arguments z1 and z2. For consistency, the sound velocity should be defined at the depth of the detector. The z positions then relate to the height above the seabed conform the standard detector geometry.

3 Event building

To efficiently use the data from the database, the data are downloaded, converted to ROOT format and written to disk using the general purpose application `JConvertDB`. The locally stored data are subsequently processed through an event builder which is implemented in application `JAcousticsEventBuilder`. The auxiliary script `JAcousticsEventBuilder.sh` does both steps in one go for a given list of run numbers. The events (`JACOUSTICS::JEvent`) are stored in a ROOT formatted output file. For the event building, the time-of-emission is estimated from the measured time-of-arrival by correcting for the propagation time. In this, the default detector geometry, the known positions of the tripods and the known sound velocity are used. Before use, duplicate data (entries where only `UNIXTIMEBASE` and `TOA_S` differ whilst the UTC times of arrival are the same), are removed.[1]

The remaining data are unambiguously associated to one of the emitters via a unique list of waveform identifiers associated to each emitter (see above). A fixed time window (`JTriggerParameters::TMax_s`) is then applied to the consecutive times of emission of the same emitter. This time window should be as small as possible but sufficiently large to accommodate the swaying of the strings and other uncertainties. If the number of measurements that are coincident within this time window exceeds a pre-set threshold, an event is triggered and written to disk. Due to the apparent strength of the signal, this threshold can be set to a large fraction (90% or so) of the total number of working receivers in the detector. In the presence of a large contribution of noise in the raw data, the efficiency as well as purity of the events will then still be very high (typically $\geq 95\%$), without the need to select data based on the quality of the measurement (i.e. `QUALITYFACTOR`).

A set of events from different emitters within a certain time window constitutes the input data for the model fit. This time window should cover the maximal time difference between the sequences of pings from the different emitters (i.e. 10 minutes). As a consequence, the complete sequence of pings from each emitter will be included anyway. For a standard data taking run with the current ORCA detector (`OID`="D_ORCA006" or `SERIALNUMBER`=49), the time to process all raw data from the database requires about 20 seconds (excluding the time to download the data from the database).

4 Model

As a start, the model should provide for a complete description of the geometry of the detector as function of time. For practicality, the geometrical description of the detector is *parametrized*. For a given string and floor number, the actual position of the optical module is then defined by a set of parameter values. A subset thereof can be classified as "fixed" parameters (see below). The values of these parameters do not change as a function of time but may need to be tuned when their accuracy is insufficient (see section 4.1). For the dynamic position calibration, a complementary subset of "free" parameters should regularly be determined from the available acoustic data. All together, the model simply comprises a list of parameter values which –via predefined methods– yield the positions of the optical modules. The free parameters and the predefined methods are defined in the class `JModel` and name space `JGEOMETRY`, respectively. The values of the free parameters are determined by a fit of the model to the acoustic data (see below). The time dependence of the detector geometry can then be reproduced by time-wise interpolation of the fitted parameter values. The current list of free parameters includes two tilt angles for each detector string and one time-of-emission for each ping. In this, the two tilt angles constitute a proxy for the two components of the sea current. Due to the sequence of pings, a complete data set usually covers 10 minutes of operation time of the detector. Assuming that the strings do not significantly sway during this time, the number of free parameters n_f thus amounts to:

[1] The problem of duplicate data is followed up in GIT issue 115 in the DAQ working group.

$$n_f = \sum_{i=1}^{M} n_i + N \times 2 \tag{2}$$

where M is the number of emitters, n_i the number of pings per sequence for emitter i and N the number of strings. Assuming that each ping is recorded in all receivers, the number of measured times of arrival (read data points) n_p amounts to:

$$n_p = \sum_{i=1}^{M} n_i \times N \times 18(19) \tag{3}$$

where the number of receivers on a string is 18, or 19 if a hydrophone is mounted on the anchor of a string.

As can be seen from equations 2 and 3, the number of degrees of freedom $\mathrm{NDF} \equiv n_p - n_f$ generally is very large, which is an asset. In the global fit, the statistical accuracy of the *a priori* unknown time-of-emission is proportional to $\sqrt{N \times 18(19)}$. As a result, the accuracy of the dynamical position calibration is proportional to square root of the size of the detector, in other words "the bigger the detector, the better the calibration". It is also interesting to note that the number of data points increases with every additional tripod by $n_i \times N \times 18(19)$ and with every additional hydrophone by $\sum_{i=1}^{M} n_i$.

4.1 Fixed parameters

The fit of the model to the acoustic data requires a number of fixed parameters. These parameters are not fitted on an event-by-event basis but are determined beforehand. For an optimal result, however, it may be required to tune these parameters (see below). The list of fixed parameters includes:

- a, b and z_0 of the sound velocity V (z_0 corresponds to the seabed);
- (x, y, z) position of the top of the T-bar of each string;
- h height of each floor in each string;
- (x, y, z) position of each emitter;
- $(\Delta x, \Delta y, \Delta z)$ relative position of the piezo sensor with respect to the floor;
- $(\Delta x, \Delta y, \Delta z)$ relative position of the hydrophone with respect to the T-bar;

The values for the sound velocity are taken from reference [1]. Estimates of the (x, y) positions of the T-bar as well as the height of the floors in the strings can be derived from the default detector geometry (e.g. "detx" file) but note that the (z) positions of the T-bars should separately be provided somehow. By definition, the height of a floor corresponds to the distance between the centre of the optical module and the top of the T-bar. The relative position of the piezo sensor with respect to the centre of the optical module is globally defined by method `getPiezoPosition()`. In this, rotational symmetry is assumed. The relative position of the hydrophone depend on the orientation of the anchor of the string. These positions are separately defined for each detector and available in ASCII formatted files named `hydrophone_DDDDDDDD.txt`, where `DDDDDDDD` corresponds to the 0-prepended serial number of the detector (i.e. `SERIALNUMBER` in table "detectors"). The positions of the tripos are also separately defined for each detector and available in ASCII formatted files named `tripod_DDDDDDDD.txt`.

The various files `XXX_DDDDDDDD.txt` are installed in directory `$JPP_LIB` by the standard `make` procedure. They will be copied to corresponding local files `XXX.txt` by executing the script `JAcoustics.sh` and providing the serial number of the detector as argument. The local files are subsequently used in various

other scripts. Some auxiliary methods are defined in the same script which are used to update one of these local files as a part of the tuning procedure presented in section 6.

4.2 Mechanical model

The tilt angles of a string correspond to its inclination with respect to the vertical at the top of the T-bar. A modeling of the shape of the full string shows that, in the presence of a buoy, the top of a string is curved towards the vertical [2]. This curvature is approximately taken into account by an *effective* height of the floor.

$$h' \equiv h + b \times (1 - \log ah) \tag{4}$$

In this, a and b are phenomenological parameters which may depend on the string but can otherwise be considered fixed. The values of a and b are bound –by definition– by the following constraints.

$$0 \leq a < 1/H \tag{5}$$
$$0 \leq b \tag{6}$$

where H is the height of the string.

5 Fit

The list of free parameters in the model includes one time-of-emission per ping and two tilt angles per string. In practice, the tilt angles are represented by the slopes $T_x = \frac{dx}{dz}$ and $T_y = \frac{dy}{dz}$. The two slopes of a string are related to its normalised inclination vector with respect to the vertical as follows.

$$\hat{T} \equiv \begin{pmatrix} T_x \\ T_y \\ \sqrt{(}1 - T_x^2 - T_y^2) \end{pmatrix} \tag{7}$$

The time-of-arrival of a signal emitted at time t_a from emitter a and detected at a receiver on string i and floor j can be expressed as follows.

$$t_{ij} = t_a + V^{-1} \left| \bar{x}_i + h_{ij} \times \hat{T}_i - \bar{x}_a \right| \tag{8}$$

where $\bar{x}_i$ correspond to the position of the top of the T-bar, h_{ij} to the height of floor j in string i and $\bar{x}_a$ to the position of emitter a. As can be seen from this equation, the free parameter t_a appears in a linear way but the free parameters T_x and T_y don't due to the evaluation of the distance between the emitter and the receiver. As a consequence, the fit of the free parameters to the data requires an iterative procedure as well as sufficiently accurate start values. A general-purpose minimiser exists in Jpp (e.g. class `JGandalf`) that can be used for the fit. The CPU time of this minimiser has a linear contribution due to the number of data points (to build the Hesse matrix) and a cubic contribution due to the number of free parameters (to invert the Hesse martix). Because the number of data points generally is much larger than the number of free parameters, the first contribution is the larger (see equations 3 and 2, respectively). Considering that each measured time-of-arrival involves only 3 free parameters (one time-of-emission and two slopes) and that the total number of free parameters n_f is much larger than that, a custom implementation of this algorithm will be faster by a factor $(n_f/3)^2$. So, a custom implementation is also available (class `JKatoomba<JGandalf>`). In the fit, the total χ^2 (sum of the normalised differences between the measured and modeled times of arrival) is minimised. To mitigate the effects of possible outliers in the data, a (Lorentzian) M-estimator is used.

For small tilt angles, the time-of-arrival can be expressed in terms of a linear dependence on all free parameters.

$$t_{ij} \simeq t_a + V^{-1}\left(D + \frac{x_i - x_a}{D}h_{ij}T_x + \frac{y_i - y_a}{D}h_{ij}T_y\right) \quad (9)$$

where

$$D \equiv \sqrt{(\bar{x}_i - \bar{x}_a)^2 + h_{ij}^2 + 2(z_i - z_a)h_{ij}} \quad (10)$$

The fit of the free parameters to the data is then strictly linear. As a consequence, it does not require start values and takes only one step (see e.g. reference [3]). The linear fit is implemented in class `JKatoomba<JEstimator>` and is used to provide start values for the iterative procedure. The complete fit procedure is implemented in application [script] `JKatoomba[.sh]`[2]. The fit results (`JACOUSTICS::JEvt`) are stored in a ROOT formatted output file. They can be monitored using application [script] `JCanberra[.sh]`.

The output data volume for a standard data taking run is very small (compared to the raw data) and linearly scales with the rate of the fits (i.e. $(10\ \mathrm{min})^{-1}$) and the number of free parameters (n_f in equation 2). For a standard data taking run with the current ORCA detector (`OID`="D_ORCA006" or `SERIALNUMBER`=49), the time to fit all events produced by the event builder takes about 5 seconds and the data volume amounts to about $1\ \mathrm{MB}$.

6 Tuning of fixed parameters

The values of the fixed parameters can be tuned as follows. For a given detector and a selection of data taking runs, the acoustic data are processed through the event builder. The values of one or more fixed parameters are then varied and for each combination of parameter values, the model is fitted to the processed data. This is referred to as a "scan". In each scan, the average χ^2 per degree of freedom of the fits is measured as a function of these parameter values. The optimal values are then determined by fitting a custom function to the average χ^2 per degree of freedom. For this tuning of the fixed parameters, a suite of scripts is available in directory `$JPP_DIR/examples/JAcoustics/`. The current list of scripts includes:

```
detector-XY:(run|plot|fit).sh
detector-Zmul:(run|plot|fit).sh
module-Z:(run|plot|fit).sh
tripod-3Z:(run|plot|fit).sh
tripod-3Z:(run|plot|fit).sh
tripod-Z:(run|plot|fit).sh
mechanics:(run|plot).sh
```

In this, one or more parameter values (listed between - and :) of some part of the system (named before -) are evaluated. The actions `run`, `plot` and `fit` correspond to the scan of the parameter values, the plotting of the average χ^2 per degree of freedom and the determination of the optimal values, respectively. Note that the scan may require a long time to execute depending on the number of evaluations to be made. For the `fit` actions related to the detector and the tripods, the local detector file ("detx") and the local tripod file (`tripod.txt`) will be updated, respectively. The latter could eventually replace the file `tripod_DDDDDDDD.txt` in directory `$JPP_DIR/software/JAcoustics/` so that the results are archived and reusable.

It should be noted that this determination of the fixed parameters does not constrain the absolute orientation of the detector. For this, external input is required (e.g. results from a study of atmospheric muons produced by cosmic rays that are shadowed by the moon).

[2] Katoomba is a small town in the Blue Mountains, Australia.

7 Application

After the tuning of the fixed parameters (see above), the available data can now be processed for the dynamic position calibration. To efficiently use the data from the database, the data are downloaded, converted to ROOT format, written to disk and subsequently processed using the scripts `JAcousticsEventBuilder.sh` and `JKatoombo.sh`.

The output can subsequently be loaded in memory using class `JDynamics`. In this, Legendre polynomial interpolations are used to provide the tilt angles of each string at a specified time.

8 Summary

A dynamic position calibration of the detector is implemented in the Jpp software framework. The overall procedure is fast and produces a small amount of data. More (technical) documentation is available in Doxygen format at `https://common.pages.km3net.de/jpp/` (see `Namespaces` → `JACOUSTICS`).

References

[1] Vincent Bertin, private communications.

[2] Edward Berbee, private communications.

[3] Statistical Methods in Data Analysis, W.J. Metzger, HEN-343