Belle II and SuperKEKB:
- KEK, Tsukuba, Japan
- Next generation B-factory
- Search for New Physics
- Study of CP violation in B-meson decay
- $e^+e^-$ asymmetric collider (SuperKEKB)
- Beam energy of $e^+ = 7$ GeV
- Beam energy of $e^- = 4$ GeV

Belle II detector:
- DEPFET pixel (PXD) sensors
- Double-sided strip (SVD) sensors
- Central drift chamber (CDC)
- Time of propagation (TOP) counter
- Aerogel ring-imaging Cherenkov (ARICH) detector
- Electromagnetic calorimeter (ECL)
- $K_L$ and muon (KLM) detector

**Design of CAF**
The calibration and alignment framework is implemented to Belle analysis and simulation framework II (b2f2), which the user controls via a Python interface. It is composed by two virtual C++ classes: collector module and calibration algorithms and Python interface is used [1].

1) **Collector module**: Collector module builds data objects for the calibration algorithms. It has two parts: preparation and collection. In the first part user define and initialize objects (trees, histograms, etc.), which are filled in the second part of collector module.

2) **Calibration algorithms**: Calibration algorithms is a base C++ class which developers inherit from to implement their own C++ Algorithm class. It analyse collected data and determine calibration and alignment constants. The output status is one of:

   - Success: the algorithm was successfully finished. Calibration or alignment constants are saved to current local database.
   - Iterate: The algorithm was successfully finished, but iteration of collector and calibration is requested. The calibration or alignment constants are saved in current local database.
   - Not enough data: the algorithm did not have enough data to successfully finish. No constants are saved.
   - Failure: the algorithm failed for any other reason. No constants are saved.

3) **User interface in Python**: To provide the user with workflow flexibility, the Python interface makes it possible to separate the collection and calibration stages and to include or exclude other related standard modules (simulation, reconstruction, etc.) and user-written modules. The Python interface permits parallel processing of data from multiple runs.

**Alignment tools**
The important function of CAF is the position alignment of the detector sensors. The modern alignment tools are used for silicon sensors (PXD and SVD) only, and they are able to clearly and precisely calculate position of wires (CDC), or muon-detector strips (KLM). The track-based alignment tools are used for silicon sensors (PXD and SVD, KLM CDC). This module is used the Milepede II algorithm (for explanation see Alignment tools).

(i) **Millepede II algorithm**

The Milepede II algorithm is tool for precise determination of alignment constants and it is based on simultaneous (linearised) minimisation of residuals with respect to all tracks and alignment parameters. No approximations is used. The linearised minimisation expects iterations of alignment and calibration algorithms, but all parameters corrections are kept from every iteration.

(ii) **General Broken Lines**

The General Broken Lines method provide fast track refitting with multiple scattering. The method introduces additional fit parameters: kink angles at predefined scattering points. A track seed is propagated in detector material to parametrize multiple scattering. A track is generated by the algorithm was successfully finished. Calibration or alignment constants are saved to current local database. If the algorithm did not have enough data to successfully finish. No constants are saved. If the algorithm failed for any other reason. No constants are saved.

**Alignment tools**
The important function of CAF is the position alignment of the detector sensors. The modern alignment tools are used for silicon sensors (PXD and SVD) only, and they are able to clearly and precisely calculate position of wires (CDC), or muon-detector strips (KLM). The track-based alignment tools are used for silicon sensors (PXD and SVD, KLM CDC). This module is used the Milepede II algorithm (for explanation see Alignment tools).

(i) **Millepede II algorithm**

The Milepede II algorithm is tool for precise determination of alignment constants and it is based on simultaneous (linearised) minimisation of residuals with respect to all tracks and alignment parameters. No approximations is used. The linearised minimisation expects iterations of alignment and calibration algorithms, but all parameters corrections are kept from every iteration.

(ii) **General Broken Lines**

The General Broken Lines method provide fast track refitting with multiple scattering. The method introduces additional fit parameters: kink angles at predefined scattering points. A track seed is propagated in detector material to parametrize multiple scattering. A track is generated by the algorithm was successfully finished. Calibration or alignment constants are saved to current local database. If the algorithm did not have enough data to successfully finish. No constants are saved. If the algorithm failed for any other reason. No constants are saved.

**Types of calibration and alignment**

1) **Online** calibration and alignment uses a small set of quickly reconstructed data. The online reconstruction uses a tailored set of quick algorithms for partial event reconstruction.

2) **Fast** calibration and alignment will determine constants required to procedure analysis quality data. It will trade accuracy for speed. Calibration and alignment will be done during converting data. Turn-around time must be less than one day from collection of the input data.

3) **Offline** or reprocessing calibration and alignment will run as many calibration algorithms as required to produce best-quality constant. Large set of data will be used.

**Reference**