Implementation of Genfit2 as an experiment independent track-fitting framework

Tadeas Bilka[1], Nils Braun[2], Thomas Hauth[2], Thomas Kuhr[3], Lia Lavezzi[4, 5], Felix Metzner[2], Stephan Paul[6], Elisabetta Prencipe[7, a], Markus Prim[2], Johannes Rauch[6], James Ritman[7, 8], Tobias Schlüter[9], Stefano Spataro[5]

[a] = corresponding author
[1] Charles University of Prague (CZ);[2] Karlsruher Institut für Technologie (DE); [3] Ludwig-Maximilians-Univesität München (DE); [4] Institute of High Energy Physics (CN); [5] Universitá di Torino and INFN (IT); [6] Technische Universität München (DE); [7] Forschungszentrum Jülich (DE); [8] Ruhr-Universität Bochum (DE); [9] LP-Research Inc. (JP)

Abstract

The Genfit toolkit, initially developed at the Technische Universität München, has been extended and modified to be more general and user-friendly. The new Genfit, called Genfit2, provides track representation, track-fitting algorithms and graphic visualization of tracks and detectors, and it can be used for any experiment that determines parameters of charged particle trajectories from spacial coordinate measurements. Based on general Kalman filter routines, it can perform extrapolations of track parameters and covariance matrices. It also provides interfaces to Millepede II for alignment purposes, and RAVE for the vertex finder. Results of an implementation of Genfit2 in basf2 and PandaRoot software frameworks are presented here.

February 14, 2019

Contents

1   Introduction

Track reconstruction is an essential part of most nuclear and particle physics experiments.

GENFIT2 [1] is an experiment-independent modular framework for track-fitting and other related tasks. The project is open-source and available on GitHub[1].

All particle physics experiments need to identify and classify processes based on detector signals. Combining these signals to recover particle trajectories is the task called "track reconstruction": suitable collections of measurements must be combined into track candidates. Tracks must be fitted, and points of common origin or exit, vertices, must be found and fitted. Track finding and track fitting are not independent from each other. Steps of different levels of track finding and refining alternate with track-fitting steps. Fitted tracks from different tracking subdetectors can be matched and combined into larger tracks; and single measurements can be appended to existing tracks, which might come from another subdetector. With provided suitable collections of measurements and detector geometry, all these tasks can be performed by GENFIT2.

GENFIT2 extends and improves the work previously performed by GENFIT [2], the external tracking tool available in the PandaRoot releases since 2009. PandaRoot [3, 4] is the official software framework of the $\overline{\text{P}}$ANDA Collaboration [5], which is building a future $\bar{p}p$ experiment investigating reactions where an antiproton beam with momentum between 1.5 GeV/c and 15 GeV/c impinges on a fixed target. $\overline{\text{P}}$ANDA, which will be located at the international FAIR facility in Darmstadt (Germany), will consist of two main sections: the target spectrometer, which will have a solenoidal magnetic field ($\mathbf{B}$ = 2 T); and a forward spectrometer, which will include a dipole field with 2 T·m maximum bending power.

The new GENFIT2 toolkit was already successfully used during a test of the Belle II high-level trigger architecture and the combined Belle II vertex detector readout architecture [6]. It is used for recorded Phase 2 data and huge MC productions. Belle II [7] is a major upgrade of the Belle detector [8], which operates in a solenoid magnetic field of 1.5 T at an e^+e^- collider in Tsukuba (Japan), running at a center of mass energy equal to that of the $\Upsilon(4S)$[2]. During the aforementioned test it served for online track reconstruction in the data reduction stage of the

---

[1]https://github.com/GenFit/GenFit.

[2]The Belle II experiment concluded the Phase 2 data taking period on July 17th, 2018. During the Phase 3, planned on spring 2019, the experiment will run at the energy in the center of mass of the $\Upsilon(4S)$ and $\Upsilon(3S)$.

high-level trigger, as well as for offline analysis. It was also used to supply input to the Millepede II alignment software [9].

This paper documents the implemented classes, how to use them, and also presents performance results with the PandaRoot and the basf2 software frameworks. The latter is the official Analysis and Software Framework [10, 11] of the Belle II Collaboration. This document is organized as follows: after this general introduction to GENFIT2, the motivation to build this toolkit is presented in the second Section. In Section 3 and 4, generalities about track parameterization, extrapolation and linearization are provided; the documentation of the GENFIT2 classes is given in Section 5, with the description of the general structure of this modular code. Section 6 describes the different track fitters available in GENFIT2 ; Section 7 provided a short description of the interface between GENFIT2 and RAVE [12]; Section 8 shows the performace in basf2 and PandaRoot; finally, the work is summarized in the Conclusion.

2   Motivation for an experiment-independent track-fitting toolkit

In order to perform an analysis it is necessary to measure the position of a particle with its uncertainty, and have a knowledge of the particle momentum.

A *measurement* is a processed electronic signal recorded by a detector, which serves as input to the reconstruction algorithms. Physics analyses generally work with particle *tracks*. *Tracks* can *e.g.* be parametrized by an Euclidean position $\vec{x}$ and momentum $\vec{p}$ with the corresponding covariance matrix. To form a track from a set of measurements, two tasks have to be performed: *track-finding* and *track-fitting*. *Track-finding* groups measurements which are likely to originate from the same particle into track candidates. This is a task on its own, and will not be detailed here. *Track-fitting* takes the track candidates as input, and computes track parameters at any given point along the track, together with the covariance matrix of the track parameters, taking in consideration effects due to particle interactions with materials and the magnetic fields.

Several high-energy-physics experiments implement their own track fitters; however, they use similar algorithms, such as the well known *Kalman filter* [13]. The idea of GENFIT2 is to provide an open-source, modular and extensible framework which is capable of performing track-fitting and other related tasks, and can easily be adapted to various experimental setups. Smaller experiments, which do not have the manpower to develop their own track fitter, or new experiments which need a working tool to do research and development, are encouraged to use GENFIT2.

3   Track parameterization and extrapolation

In the absence of interaction, the trajectory of a particle with charge q in a magnetic field $\mathbf{B}$ is determined by the equations of motion, given its momentum vector $\mathbf{p}$, in a single point along the track. In order to describe a track, its trajectory has to be parameterized. In a general geometry, and inside a magnetic field, it is most useful to use the distance along the track s as free parameter, and then give the values:

$$(q/p, u, v, u', v') \tag{1}$$

where u, v are the rectangular coordinates of a plane which the track intersects at distance s, and u', v' are the direction cosines, *i.e.* the projections of the direction of momentum on the

coordinate axes. If the plane is that of a planar detector, it is most useful to have u (and v) coincide with the coordinate(s) measured by the detector.

GENFIT2 extrapolates tracks using the standard Runge-Kutta-Nyström method [14] as modified by Bugge and Myrheim to carry along the Jacobian matrix [15, 16]. The code shares the same pedigree as the STEP extrapolation code used in the ATLAS experiment [17, 18]. The corresponding class in GENFIT2 is `RKTrackRep`.

During the Runge-Kutta extrapolation, global track parameters are used, namely q/p, the spatial coordinates $\boldsymbol{x} = (x, y, z)^T$ and the unit tangent vector pointing along the track, *i.e.* the derivative of $\boldsymbol{x}$ with respect to the track length s.

In order to reduce bias of the final fit result due to inaccurate seed values, which can lead to failed linearization of the Kalman filter, usually several fitting steps are done until the fit converges. For example, we can assume here that the track is *e.g.* processed three times in forward and backward directions, giving an accurate estimation of the track parameters at the starting position of the track. The last backward update of one iteration is usually taken as a starting value for the next iteration. In order to minimize the bias of the starting values, their covariance has to be very large. However, if it is too large, numerical problems can arise. It is practical to multiply the covariance of the starting value by a factor of 500 up to 1000 between 2 iterations. This factor can of course be adjusted by the user, depending on the specific case.

The fitter has a minimum and maximum number of iterations, which are assigned default values of 2 and 4, respectively, or the user can assign other values. As soon as the minimum number of iterations has been reached, the tool checks if the p-value has changed by less than a certain amount with respect to the previous iteration. The default value to check if the fit should perform a next iteration is set to 10^{-3}. However, tracks with a p-value close to zero are often considered as converged with this criterion, while the χ^2, albeit big, is still changing significantly, indicating that the fit is still improving. This situation occurs often for tracks which are given bad seed values. Thus, a non-convergence criterion has been introduced: if the relative change in χ^2 from one iteration to the next is more than 20%, the fit will continue. The value of this non-convergence criterion can be defined by the user.

After the extrapolation has reached the target surface, global coordinates are converted to local coordinates (see Eq. (1)), and the Jacobian for the extrapolation is calculated from the Jacobian matrices of the extrapolation steps J_i together with those of the coordinate changes

local→global on the starting plane A, and global→local on the target plane B as:

$$F_{A \to B} = J^B_{\text{global} \to \text{local}} J_n J_{n-1} \cdots J_1 J^A_{\text{local} \to \text{global}}. \tag{2}$$

More details about the global→local coordinate system transformation are given in the next Sec. 4 and Appendix A.

4   Linearization

All implemented track fitting algorithms use a discrete model, where the particle travels from detector plane A to detector plane B such that the parameters on plane B are given as a function of the parameters on plane A, *i.e.* $\boldsymbol{p}_B = h_{A \to B}(\boldsymbol{p}_A)$, and the Jacobian of this transport is the derivative of this function. Following the notation in Ref. [27], we use the symbol:

$$F_{A \to B} \equiv \left. \frac{\partial h_{A \to B}(\boldsymbol{p})}{\partial \boldsymbol{p}} \right|_{\boldsymbol{p}=\boldsymbol{p}_A}.$$

The covariance of the track parameters on plane B can then expressed as:

$$C_B = F_{A \to B} C_A F^T_{A \to B} + N_{A \to B}, \tag{3}$$

where $N_{A \to B}$ is the noise matrix evaluated by the track extrapolation algorithm. The evaluation of $(\boldsymbol{p}_B, C_B)$ is referred to as prediction.

Each measurement m_i gives rise to a residual r_i, the difference between the measurement expected based on the track parameters $\boldsymbol{p}_i$ and the measured value, which in general is given as:

$$r_i = m_i - f_i(\boldsymbol{p}_i), \tag{4}$$

where f is the measurement function which calculates the expected measurement given a set of track parameters. The measurement itself has a covariance matrix V_i. Since our track parameterization uses planar coordinates, this can be brought into a simple linear form when the u (and v for two-dimensional detectors) direction and zero point of the plane - in which the track parameters are given - coincide with that of the measured coordinates,

$$r_i = m_i - H_i \boldsymbol{p}_i. \tag{5}$$

In this way, no further linearization is needed in order to apply the Kalman filter to this measurement model, and H_i is a constant projection matrix for each detector. In general, H_i has to be replaced with the derivative of f_i.

The evaluation of $F_{A\to B}$ and $N_{A\to B}$ on the other hand cannot be eliminated by such a judicious choice of coordinates, and its value depends on the unknown track parameters. Especially the first few hits pose a problem: the track parameters only become determined successively while the transport matrix already needs to be evaluated.

GENFIT2 has two approaches to this problem:

- use the current result of the track fit, which may lead to imprecise or false results (it is the `SimpleKalmanFitter` described in Sec. 6);

- use information from the track finders to estimate an initial set of transport and noise matrices, then after each iteration of the fit prepare a new set of matrices which is used in the next iteration (it is the `ReferenceKalmanFitter` described in Sec. 6).

5   Structure of the code

GENFIT2 is a C++ object-oriented modular tool, structured as shown in Fig. 1. It provides general track-fitting tools, which are based on three pillars:
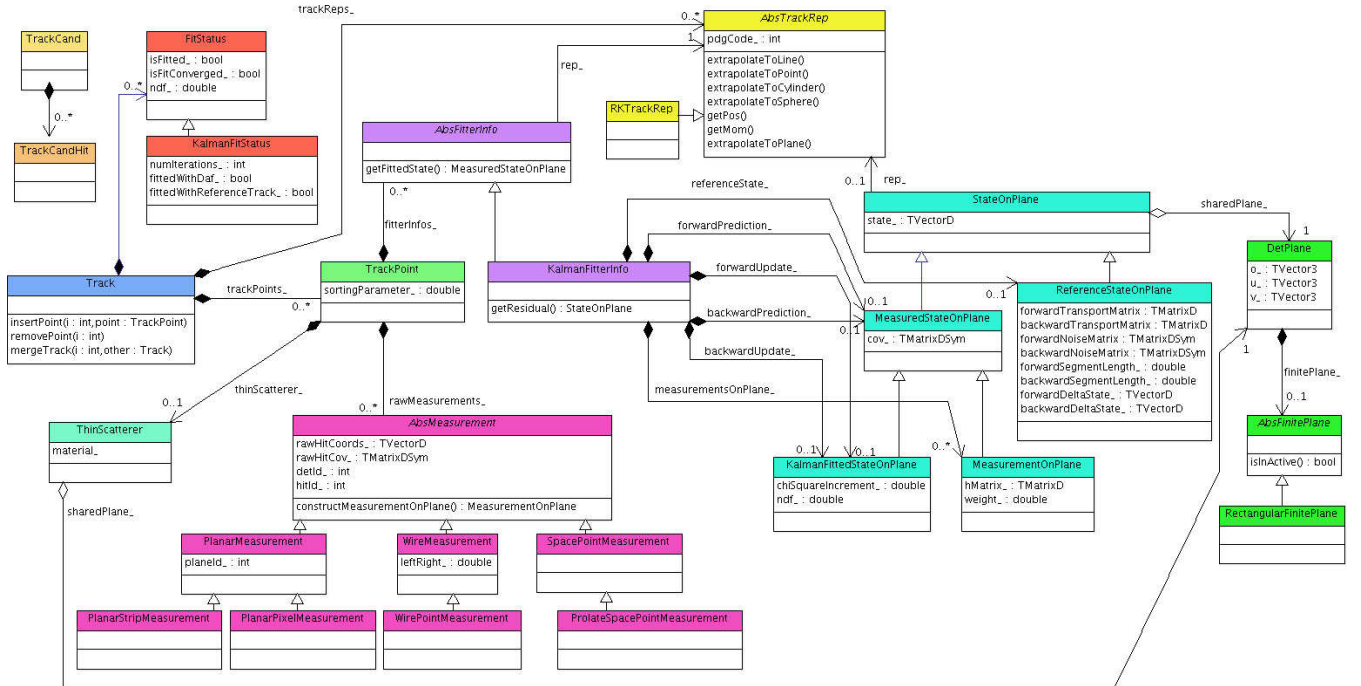

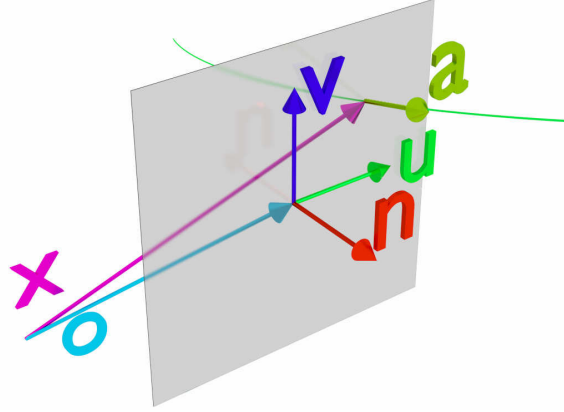Figure 1: Structure of the GENFIT2 package [1].

Figure 2: Track parametrization in the GENFIT2 class `RKTrackRep`.

- measurements;

- track representations;

- fitting algorithms.

Measurements serve as objects containing measured coordinates from a detector. They provide functions to construct a (virtual) detector plane and to provide measurement coordinates and covariance in that plane. The abstract base class `AbsMeasurement` defines the interface. A measurement can be one-dimensional (*e.g.* a line segment measured by a semiconductor strip detector), two-dimensional (*e.g.* a position on a semiconductor pixel detector or a drift isochrone of a wire detector), or three-dimensional (*e.g.* a point in space measured by a Time Projector Chamber (TPC)). GENFIT2 comes with predefined measurement classes for various detector types, which are called `PlanarMeasurement`, `WireMeasurement` and `SpacePointMeasurement`.

In GENFIT2, state vectors $\mathbf{p}$ and measurements m are given in local coordinate systems (see formulae in Appendix A), which are defined by detector planes and implemented in the `DetPlane` class. A detector plane has an origin $\vec{o}$, and is spanned by two orthogonal unit vectors $\vec{u}$ and $\vec{v}$. The normal vector $\vec{n}$ is then equal to $\vec{u} \times \vec{v}$ (see Fig. 2). Measurements provide a matrix that can project the state $\mathbf{p}$ and covariance C into the measurement coordinate system of m and its covariance V. For the Kalman filter mathematics, which are detailed in Appendix A, the state $\mathbf{p}$ and covariance C have to be projected into the measurement coordinate system of m and V. These projections are described by matrices, as in Eqs. 6-8.

If we consider a silicon strip detector with strips parallel to $\vec{u}$, the coordinate v in the local

plane system reads as:

$$m = (v).$$ (6)

The H matrix now simply has to project out the v component of p:

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$ (7)

Now, Hp is the projected state, and HCH^T is the projected covariance.

For a detector which measures u and v, the following applies:

$$m = \begin{pmatrix} u & v \end{pmatrix}$$ (8)

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$ (9)

For planar detectors, the detector plane is given by the detector geometry, whereas for `wire` and `spacepoint` measurements, so-called virtual detector planes are constructed by extrapolating the track to the point of closest approach (POCA[3]) to the spacepoint or wire. Additional detectors effects can be used here, allowing one to compensate for detector deformations like plane bending, wire sagitta, and misalignment during this procedure.

Drift-time corrections and cluster fits (track-dependent clustering) are possible.

5.1   Track data structure

All per-track data is kept in the *Track* object. It holds a set of `TrackPoint` objects, which can contain *Measurements*, *FitterInfo* objects, which hold all fitter-specific information, and thin scatterers (currently only used in the GBL fitter, see Section 6.3).

A *Track* contains one or more track representations, representing the particle hypotheses that should be used for the fit. One of them must be selected as the cardinal track representation. This can be done by the user or by GENFIT2, which selects the track representation that best fits the measurements (*e.g.* it has the lowest χ^2). The *Track* also contains a *FitStatus* object, which stores general information (number of iterations, convergence, etc.) and fit properties (χ^2, NDF, p-value, track length, etc.). The track candidate (`TrackCand`) serves as a helper class, basically storing indices of raw detector hits in *TrackCandHit* objects, which can also be overloaded by the user to store additional information. *WireTrackCandHit* objects can store the left-right ambiguity.

---

[3]POCA stands for Point of Closest Approach.

A possible use case for this possibility to incorporate within these objects is the pattern recognition: algorithms can work on raw objects that have a lighter weight content than a *Measurement*, or implement other features needed by the pattern recognition, but not necessarily by GENFIT2.

5.2 Pruning

The GENFIT2 class `MeasurementFactory` can build a *Track* from a `TrackCand`. This *Track* can then be processed by the various fitting algorithms. After fitting, *Track* objects contain a lot of data: track representations, `TrackPoint` objects with measurements, *FitterInfo* objects, etc. Usually, not all of this information needs to be stored on disk. For example, for physics analysis only parameters related to the vertex and information on the track quality are necessary. The user can decide which data to keep, *e.g.* only the fitted state of the first `TrackPoint`, and only for the cardinal track representation.

5.3 Handling of multiple measurements in one TrackPoint

Wire detectors measure a drift isochrone (a cylinder, in the simplest case). Intersected with the detector plane, this gives two (one dimensional) measurements, since the particle could have passed on either side of the wire. GENFIT2 provides the possibility to store several measurements of the same type in one `TrackPoint`[4]. These tracks can also be fitted with the Kalman filter, therefore GENFIT2 provides several options how to handle multiple measurements:

- *average*: the average of the individual measurements is calculated. This option is primarily used for the `DAF` (see Sections 6.1 and 6.2).

- *closest to prediction*: the measurement which is closest to the state prediction is selected.

- *closest to reference*: the measurement which is closest to the reference track is selected. This can only be used with the Kalman filter with reference track.

- *closest to prediction/reference for wire measurements*: if the `TrackPoint` has one *WireMeasurement*, select the side which is closest to the prediction/reference. Otherwise, it makes use of the *average*.

---

[4]This feature allows the DAF to assign weights to the `TrackPoint`. More information related to the DAF are in Sec. 6.2.

For wire measurements, it turned out that selecting the side which is closest to the state predictions is often the best option. The average weigthed measurement m and its covariance V are calculated from the individual measurements i as follows:

$$V = \left(\sum_i \left(w_i V_i^{-1} \right) \right)^{-1}$$
$$m = V \cdot \sum_i \left(w_i V_i^{-1} m_i \right)$$

(10)

5.4   Runge-Kutta track representation

The Runge-Kutta track representation (`RKTrackRep`) is based on a Runge-Kutta extrapolator adapted from Geant3 [19]. An abstract interface class interacts with the detector geometry. Implementations using the Root class `TGeoManager` [20] and Geant4 class `G4Navigator` [21] are available.

The state **p** is parametrized with 5 coordinates in a local plane coordinate system, as already shown in Fig. 2. The cartesian position $\vec{x}$ and direction $\vec{a}$ translate into plane coordinates according to the following equations:

$$\mathbf{p} = \left(q/p, u', v', u, v \right)^T$$
$$u' = \frac{\vec{a} \cdot \vec{u}}{\vec{a} \cdot \vec{n}}$$
$$v' = \frac{\vec{a} \cdot \vec{v}}{\vec{a} \cdot \vec{n}}$$
$$u = (\vec{x} - \vec{o}) \cdot \vec{u}$$
$$v = (\vec{x} - \vec{o}) \cdot \vec{v}.$$

(11)

where q is the charge of the track, and **p** its momentum. During fitting, material properties are used to calculate the following effects: energy loss and energy-loss straggling for charged particles according to the Bethe Bloch formula (the code is based on the code ported from Geant3); multiple scattering (according to Ref. [22] or using the Highland-Lynch-Dahl formula [23]), where the full noise matrix is calculated; and soft Bremsstrahlung energy loss and energy-loss straggling for e^- and e^+ (the code is based on the code ported from Geant3). The step sizes used for the Runge-Kutta extrapolation should be as large as possible to avoid unnecessary computation, while still being small enough to keep errors reasonably small. An adaptive step-size calculation is done in the `RKTrackRep`, taking field inhomogeneities and curvature into account. To calculate material effects correctly, extrapolation stops at material boundaries, and the step size is limited

so that a maximal relative momentum loss in the material is not exceeded. `RKTrackRep` provides different methods to find the POCA of the track to non-planar measurements. These are used to construct virtual detector planes:

- *extrapolate to point* finds the POCA of the track to a given spacepoint. The virtual detector plane contains the spacepoint and the POCA, and it is perpendicular to the track direction. A weight matrix can be used as a metric, defining the space in which the POCA will be calculated. By default, the inverted 3D covariance of a spacepoint measurement is used as a metric, which gives correct fitting results also for spacepoints with arbitrary covariance shapes. Again, the virtual detector plane contains the spacepoint and the POCA, and it is perpendicular to the track direction in the space defined by the metric;

- *extrapolate to line* finds the POCA of the track to a given line or wire. The virtual detector plane contains the line and the POCA. This routine is used for fitting wire measurements.

The intersection of the virtual detector plane and the measurement covariance gives the covariance in the plane.

`TrackRep` combines track parameterization and track extrapolation code. The state can be extrapolated through material and the magnetic field, to:

- detector planes;

- the POCA to points, lines, cylinders, and cones;

- or by a certain distance.

The state and covariance can be converted to/from a state defined in the global cartesian coordinate system, $(\vec{\mathbf{x}} \quad \vec{\mathbf{p}})$, as detailed in Appendix A. The abstract base class `AbsTrackRep` defines the interface, and GENFIT2 ships with the `RKTrackRep` as a concrete implementation.

6   Fitter implementation

Track representations combine track-parameterization and track-extrapolation code. Fitting algorithms use the measurements and track representations to calculate fit results, which are stored in corresponding objects in the *Track* objects.

Four different track-fitting algorithms are currently implemented in GENFIT2:

- two smoothing Kalman filters, one which linearizes the transport around the state predictions, and one which linearizes around a reference track;

- a deterministic annealing filter (`DAF`);

- a general broken lines (GBL) fitter.

GENFIT2 provides the possibility to store several measurements of the same type in one `TrackPoint`, mainly for using the DAF to assign weights to them. Wire measurements also produce two `MeasurementOnPlane` objects, representing the passage of the particle on either side of the wire. These tracks can also be fitted with the Kalman filters.

6.1 Kalman filter

The Kalman filter is a progressive algorithm that successively adds the information from each detector hit to the estimated track parameters using a linear model of measurement and track propagation. It uses a series of measurements observed over time, containing "noise" (random variables) and other inaccuracies, and produced estimates of unknown variables; then iterates a least square method in 3 steps: prediction, update, and smoothing.

Smoothing [26] is a standard technique: the track is fitted in forward and backward direction, where predictions and updates are saved in the $FitterInfos$ at each `TrackPoint`. With both of these, smoothed track states can be calculated. The weighted average between forward and backward prediction gives the unbiased state, whereas the average between prediction in one direction and update of the opposite direction results in the biased smoothed state. This smoothed track gives more accurate states than either forward or backward updates alone.

The Kalman filter method introduced in GENFIT2 is widely documented. We refer specifically to Ref. [27], and here we only document the specifics of our implementation which deviates in two ways. First, all matrix operations are performed in a way which explicitly maintains the symmetry of the covariance matrices at each step in the calculation, thus minimizing the number of operations and avoiding inconsistencies due to round-off errors. Second, an alternative implementation is available which implements a square-root filter which additionally guarantees that the covariance matrices remain positive-definite while improving the robustness against round-off errors or vastly different matrix entries which can especially happen at the beggining of the track fit, where individual track parameters only become known successively.

The Kalman filter performs a succession of fits (called "updates"), moving from the innermost hit along the series of hits identified by the track finders. Then the result of this first iteration of fits in the outwards direction is used as the starting value for the next iteration in which the fit proceeds in the backwards direction. In order to avoid biasing the inwards fit, the covariance matrix of the result of the outwards fit is multiplied by a large factor and the off-diagonal entries are discarded. At each `TrackPoint` the results from inward and outward fit are combined by averaging the updated state of the outward fit and the prediction of the inward fit. This averaging is called smoothing, but it should be kept in mind that also for the smoothed track, the track parameters change discretely at each measurement. Averaging is thus an important operation in GENFIT2, and some care has gone into a numerically stable, efficient implementation. This implementation is detailed in Appendix B.

Given the track parameters $\boldsymbol{p}_{i-1|i-1}, C_{i-1|i-1}$ at `TrackPoint` $i-1$ after the preceding update step, we evaluate the prediction at plane i:

$$\boldsymbol{p}_{i|i-1} = h_{(i-1)\to i}(\boldsymbol{p}_{i-1|i-1}), \tag{12}$$

$$C_{i|i-1} = F_{(i-1)\to i}C_{i-1|i-1}F_{(i-1)\to i}^T + N_{(i-1)\to i}, \tag{13}$$

and the residual of the prediction:

$$r_{i|i-1} = m_i - H_i\boldsymbol{p}_{i|i-1}. \tag{14}$$

Taking care to maintain the symmetry properties of the matrices, we can then write the updated state and covariance as:

$$\boldsymbol{p}_{i|i} = \boldsymbol{p}_{i|i-1} + C_{i|i-1}H_i^t(V_i + H_iC_{i|i-1}H_i^T)^{-1}r_{i|i-1}, \tag{15}$$

$$C_{i|i} = C_{i|i-1} - C_{i|i-1}H_i^T(V_i + H_iC_{i|i-1}H_i^T)^{-1}H_iC_{i|i-1}. \tag{16}$$

This summarizes the formalism of the Kalman filter. The only difference between the so-called `SimpleKalmanFitter` and the `ReferenceKalmanFitter` lies in the evaluation of the transport and noise matrices, as explained above. Nevertheless, we have implemented an alternative equivalent formalism, using square-roots of covariance matrices to further improve numerical accuracy and resilience. This implementation is described in Appendix C. Employing it allows significant increases in the blow up factor applied to the covariance matrix on fitter returns.

During the first iteration, the reference track is constructed by extrapolating the seed state from measurement to measurement. In the following iterations, the smoothed states of the

previous iterations are taken as a new reference states. A χ^2 value is calculated according to:

$$r_k = \mathbf{p}_{k|n} - \mathbf{p}_{k,r}$$
$$\chi_k^2 = \sum_i \frac{(r_{k|n,i})^2}{V_{k|n,i,i}}. \tag{17}$$

If the χ^2 is below a certain value (default assigned value is 1), the corresponding reference state will not be updated to save computing time.

The same convergence criteria as for the standard Kalman filter are used. Additionally, the fit converges if the reference states do not change anymore, according to Eq. [17].

The choice of the linearization point can significantly affect the performance of the track fit. In particular, if the first few hits lead to a large misestimate of the curvature, state predictions may stay very far from the actual trajectories. Consequently, linearizing around them is not optimal; material and magnetic field lookups are also not done at the proper places. It is therefore common to linearize around reference states [24], which are calculated by extrapolating the start parameters to all `TrackPoint` objects. At later iterations, the smoothed states from the previous iterations are used as linearization points. However, if the change would be very small, reference states are not updated, saving computing time. It is also possible to let the fitter sort the measurements along the reference track, which can improve the fitting accuracy. In addition to the convergence criteria detailed above, the fit is regarded as converged if none of the reference states has been updated since the previous iteration.

6.2   Deterministic annealing filter

The deterministic annealing fitter (DAF) [27] is a powerful tool for the rejection of outlying measurements. It is a Kalman filter that uses a weighting procedure between iterations based on the measurement residuals to determine the proper weights. The user can select which of GENFIT2 two Kalman filter implementations should be used, and specify the annealing scheme.

The DAF is also perfectly suited to resolving the left-right ambiguities of wire measurements. However, a problem can occur: the weights of the `MeasurementOnPlane` objects must be initialized. The basic solution is to initialize both left and right measurements with a weight of 0.5. Effectively, the wire positions are taken as measurements in the first iteration, and their covariance is twice the mean of the individual covariances. So, all the wire positions have the same covariance, no matter how far from the actual trajectory they are. This systematically false

estimate of the covariances biases the fit. GENFIT2 implements a novel technique to initialize the weights that improves the fitting efficiency: measurements with larger drift radii are assigned smaller weights, leading to larger covariances since the wire position is expected to be farther away from the trajectory. The mathematical espression used in the code is:

$$w = \frac{1}{2} \left(1 - \frac{r_{\text{drift}}}{r_{\text{drift, max}}} \right)^2,$$

(18)

where w is the given weight, and r_{drift} is the distance of the measurement from trajectory.

In contrast, measurements with smaller drift radii, which are closer to the trajectory, get larger weights. When the annealing scheme has been passed, convergence is checked: if the absolute change of all weights is less than 10^{-3} (user configurable), the fit is regarded as converged. Otherwise another iteration is done, until the fit is converged or a maximum number of iterations is reached.

6.3 Generalized broken lines fitter

The generalized broken lines (GBL) method of track fitting [28, 29] was implemented especially for the purpose of alignment with Millipede II [30]. It is mathematically equivalent to the Kalman filter (with thin scatterers instead of continuous materials), and works with a large number of parameters, but fits tracks in their entirety in one step. For alignment purposes, GENFIT2 also provides a set of interfaces for alignment parameters and derivatives which can be implemented by the detector classes.

7 Vertex reconstruction

GFRave, an interface to the vertex-fitting framework RAVE [12], has been implemented in GEN-FIT2. RAVE is a detector-independent toolkit for vertex reconstruction, originally developed for the CMS experiment [31]. GFRave takes full advantage of the GENFIT2 material model and the sophisticated algorithms of RAVE, allowing for precise and fast vertex reconstruction. GFRave is part of the basf2 framework, where it is actively used. However, there is still no corresponding implementation in PandaRoot, which is using for the time being other vertex and kinematic fitters.

8   Performance

8.1   Application of the GENFIT2 standalone toolkit

Examples of what can be the result of a standalone GENFIT2 simulation are given in Figs. 3-4, which display the screenshots of an event reconstructed with GENFIT2, performed by an algorithm based on `Teve`: the reader can see that many features are available in the left-menu. Figures 5, 6, 7 show tests performed with a demonstration macro provided together with the GENFIT2 toolkit (*makeGeom.C*). By default, this test macro creates a geometry which comprises a 1 m^3 cube filled with air, and 4 layers of cylindrical silicon material with a radius of 1.0 cm, a length of 10 cm and a wall-thickness of 0.05 cm. The creation of *measurements* in the test macros is independent on the geometry, but the geometry is basically provided to evaluate material effects. One million events with a single pion in a solenoidal field of 2 T have been simulated. Different pion momenta have been simulated: 50 MeV/c, 100 MeV/c, 400 MeV/c, 1 GeV/c. The information illustrated in the standalone GENFIT2 event visualization is related to the detector, hits, track with errors and track markers, the reference track, the result of the fit in forward and backward direction.

8.2   Visualization of tracks in basf2

GENFIT2 features a sophisticated 3D event display, which allows to visualize fitted tracks. Detector geometry, measurements, detector planes, reference tracks, forward and backward fits (predictions and updates), smoothed tracks, and covariances of measurements and tracks can be drawn. Tracks can be refitted with different algorithms and settings, and fit results can be viewed instantly. Examples have been shown in Figs. 3-4.
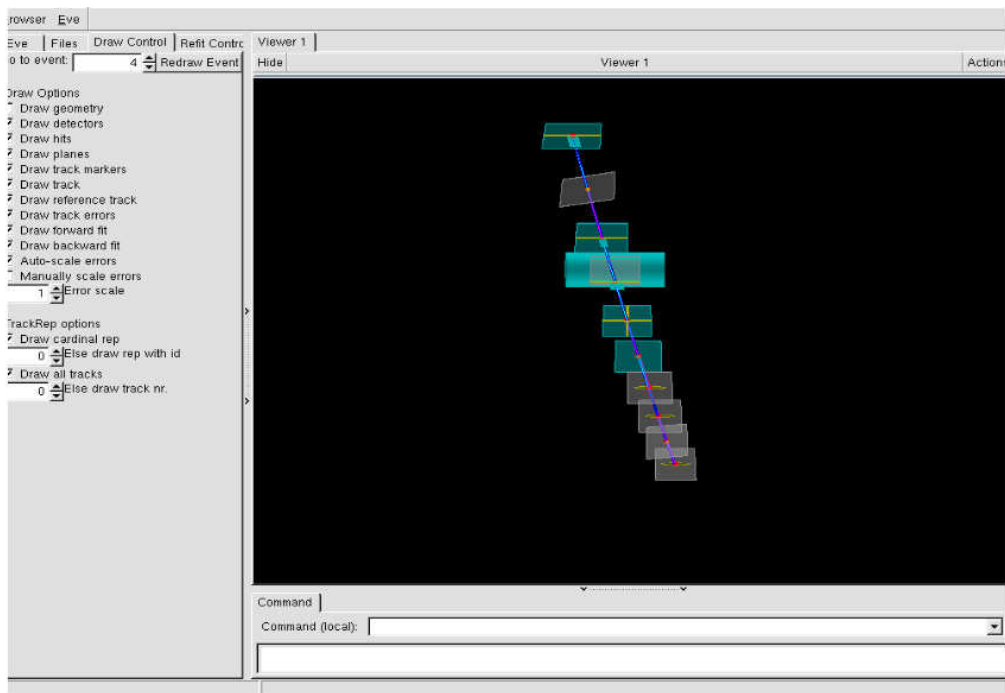
Figure 3:   Example of event visualization using the standalone GENFIT2 tool.   Here a pion of 400 MeV/c is generated in a solenoidal magnetic field **B**= 2 T.

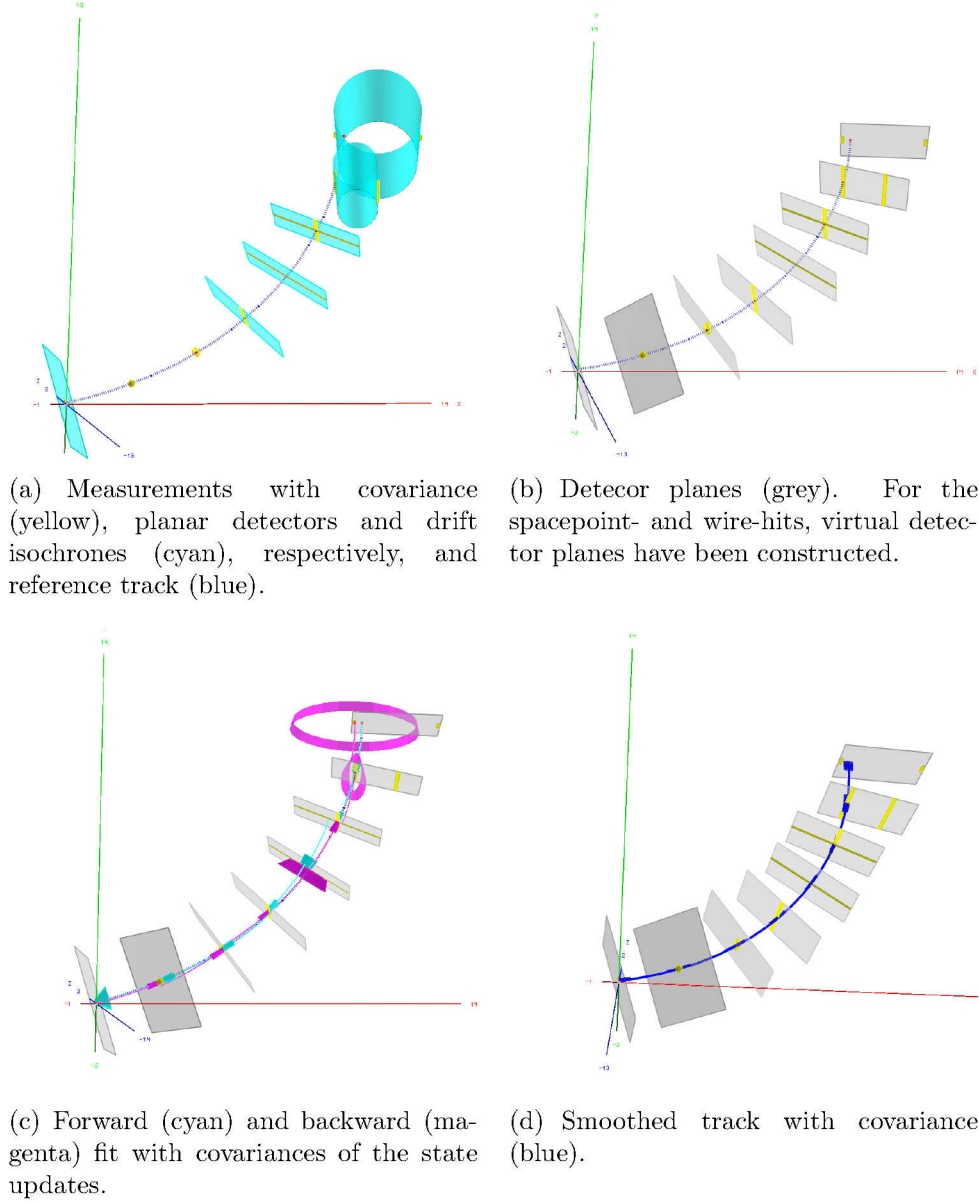(a) Measurements with covariance (yellow), planar detectors and drift isochrones (cyan), respectively, and reference track (blue).

(b) Detecor planes (grey). For the spacepoint- and wire-hits, virtual detector planes have been constructed.

(c) Forward (cyan) and backward (magenta) fit with covariances of the state updates.

(d) Smoothed track with covariance (blue).

Figure 4: GENFIT2 event display screenshots [32]. The fit of a set of hits with the Kalman filter with reference track is shown. For demonstration purposes, the different hit types supported by GENFIT2 are used, starting from the origin: planar pixel hit, spacepoint hit, prolate spacepoint hit, two perpendicular planar strip hits, double sided planar strip hit, wire hit, wire hit with second coordinate measurement.
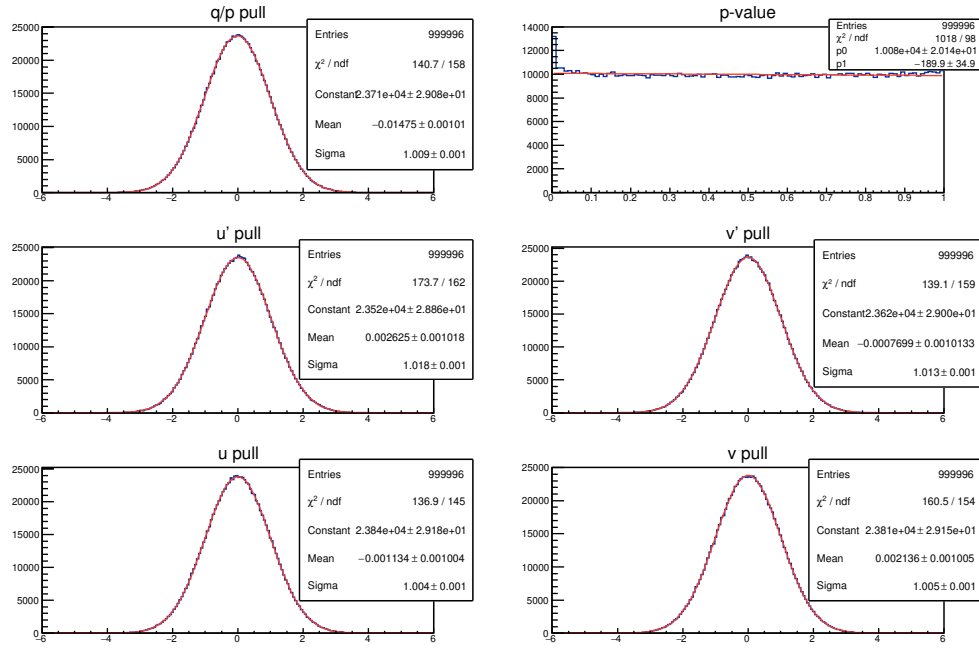
Figure 5: Pull of the track parameters $(u, v, u', v'$ and $q/p)$, with the pion momentum $=$ 1 GeV/c. Gaussian fits to these pull distributions are consistent with the expectation of a normal distribution.
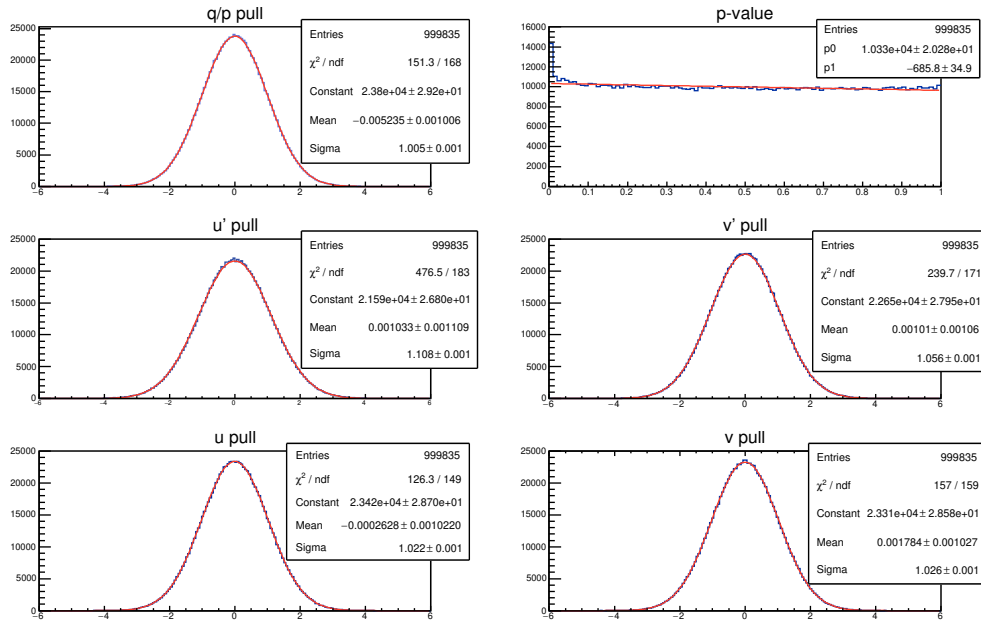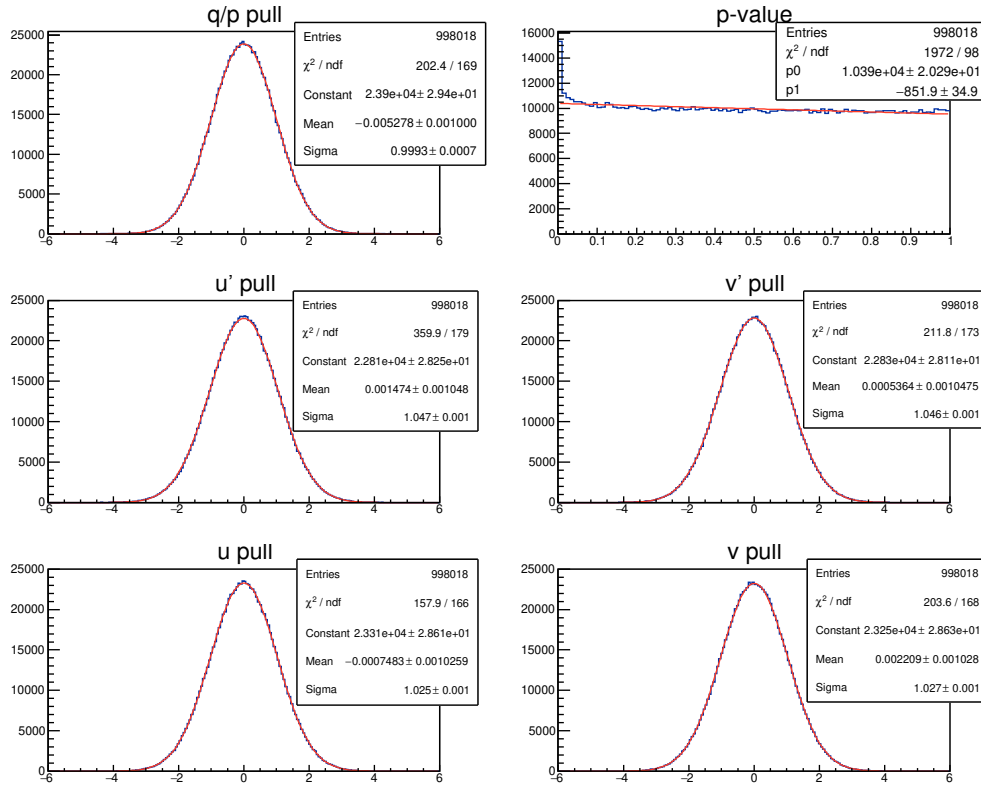
Figure 6: Pull of the track parameters $(u, v, u', v'$ and $q/p)$, with the pion momentum = 100 MeV/c. Gaussian fits to these pull distributions are consistent with the expectation of a normal distribution.

Figure 7: Pull of the track parameters $(u, v, u', v'$ and $q/p)$, with the pion momentum = 50 MeV/c. Gaussian fits to these pull distributions are consistent with the expectation of a normal distribution.
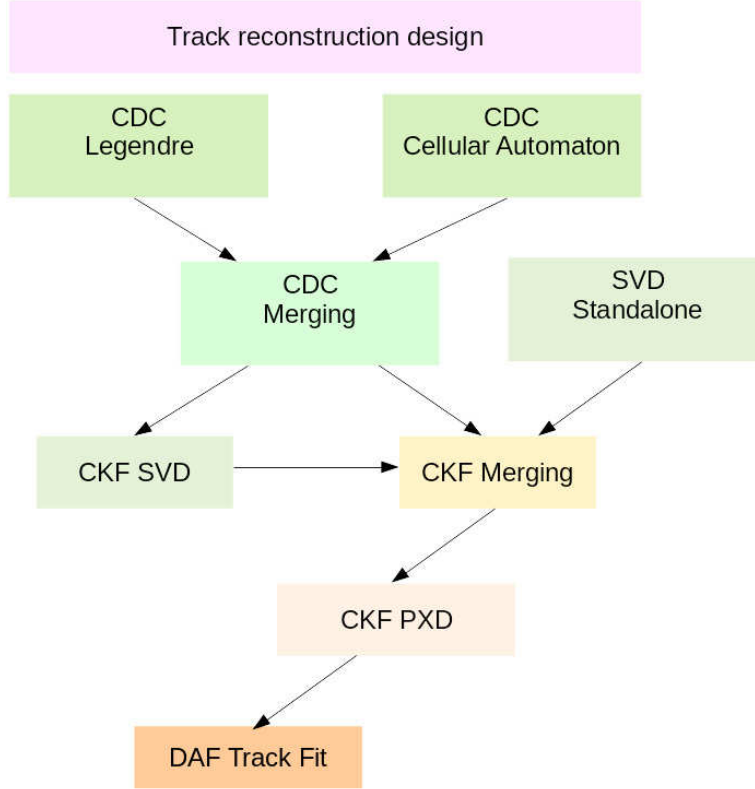
8.3   Performance of GENFIT2 in basf2

Figure 8: Scheme of the track reconstruction design in basf2 [33].

GENFIT2 performance depends on many parameters, such as computer hardware and compiler, number and types of measurements, momentum, geometry, magnetic field, number of iterations, convergence settings, and so on. Nevertheless, a study has been conducted in both, basf2 and PandaRoot, to give the reader an idea of the performance expected in high energy physics using GENFIT2. Figure 8 shows the track reconstruction design in Belle II. The Belle II detector is structured with an internal vertex detector (VXD), which is made by 2 inner layers of silicon Pixel Detector (PXD) and 4 double-side strips of Silicon Detector (SVD); then a Central Drift Chamber (CDC) is given for tracking purposes and particle identification. A time-of-propagation (TOP) detector is also furnished, surrounded by an electro-magnetic calorimeter (EM), while the outer detector is for μ and K_L^0 detections. A Combinatorial Kalman Filter (CKF) was implemented at software level for track finding issues. GENFIT2 takes care of track

fitting in basf2: it acts in a way to bind together track segments and information of PXD (planar hits), SVD (position along strips), and CDC (wire and drift time). The execution time of the basf2 `GenFitter` module has been measured on a current 3.4 GHz office PC in single threaded operation, an Intel machine with 4 cores. All code has been compiled with `-O3` optimization settings.

The module performs the fitting and a few more other features, like producing the GEN-FIT2 track from a `TrackCand`, storing the fitted track in an output array, creating relations etc., resulting in an overhead of a little less than 1 ms. Tracks are generated with a particle gun, with $\theta = 100^o$ and a momentum of 0.9 GeV/c in a constant magnetic field equal to 1.5 T. The resulting tracks have 72 ± 2 `TrackPoints`. GENFIT2 is configured to do from 3 to 10 iterations with default convergence criteria. From the results shown in Tab. 1, one can see that the *Kalman Fitter with reference track* needs less iterations to converge. Material lookup takes around 2.2 ms per iteration for the Kalman filter. The *Reference Kalman fitter* is also faster here, since the reference states are not recalculated if they are close to the smoothed states of the previous iteration. This is also the reason why the *Reference Kalman fitter* can often finish after 2 iterations. The reference track is already so close to the smoothed track, that it would not change in subsequent iterations. Table 1 shows that the calculation of the material effects takes a large proportion of the computing time (of course, it strongly depends on the complexity of the detector geometry under exam), and the Kalman fitter looks reasonably fast.

Table 1: Execution time of the GenFitter module in basf2.

| Fitter	no material effects	with material effects	n. iterations
Kalman	3.4 ms	10 ms	3
Reference Kalman	4.0 ms	8.2 ms	2
DAF	7.0 ms	11 ms	4

Examples of the good performance of GENFIT2 in basf2 are shown in Figs. 9-10: in the former a study with cosmics is performed, showing the good shape of the track reconstruction software code; in the latter a study showing the efficiency of tracks, reconstructed with basf2, is shown. The pull distributions of those parameters, defined as the ratio between the difference of reconstructed and generated values, and the error of such parameter, is parameterized by a double Gaussian function, and displayed in Fig. 9.
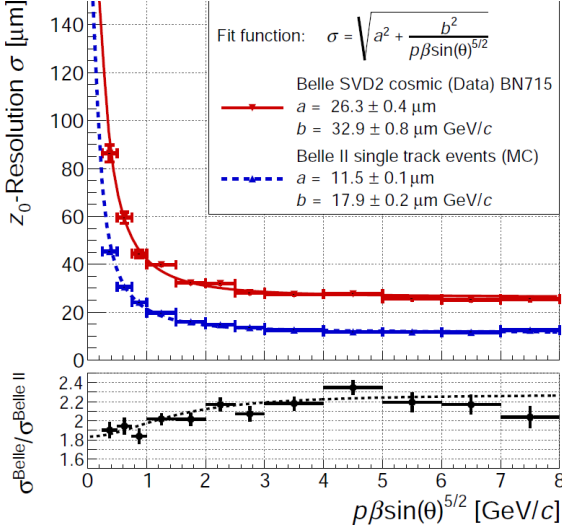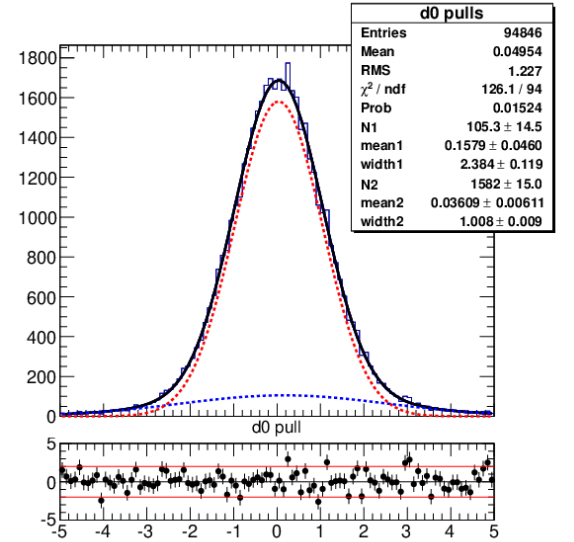
(a)
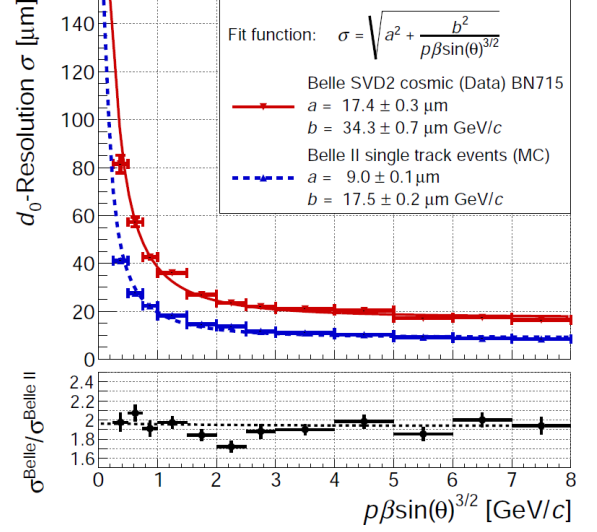
(b)
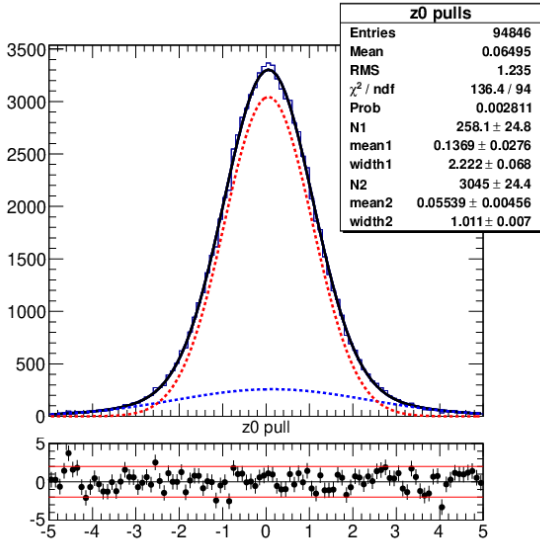
(c)

(d)

Figure 9: Study of track reconstruction in Belle II, using cosmics collected during February-March 2018: z_0 (a) and d_0 (b) resolution vs momentum. z_0 and d_0 are defined as the projections of the z axis of the POCA, and that of the x-y plane of the POCA, respectively. Fits to real data (cosmics, red plot) and to the MC simulated single track events (blue plot) are displayed. Figures (c) and (d) show respectively the fits to the pull distributions of those parameters [34]-[35].
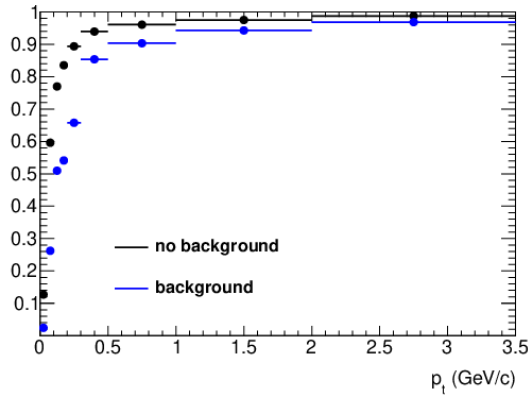
Figure 10: Efficiency of charged particles reconstructed with GENFIT2 in basf2 [34], with and without machine background simulation.

8.4 Performance of GENFIT2 in PandaRoot

An interface has been built between GENFIT2 and PandaRoot: `GenfitTools`. The basic structure of `GenfitTools` is the same used to interface the previous GENFIT package, but with a few modifications due to the new classes introduced in GENFIT2 (see Fig. 11). The new `GenfitTools` structure has been created to allow both, GENFIT and GENFIT2, to be used as fitting tool, and then compare their results.

`GenfitTools` includes several objects: *adapters*(2); *recohits*(2); *recotasks*(2); *trackrep*.

The objects *adapters*, *recohits*, *recotasks* refer to the GENFIT interface to the PandaRoot classes, and the objects *adapters*2, *recohits*2, *recotasks*2 refer to the GENFIT2 interface to the PandaRoot classes. *PndTrack* is the standard object for PandaRoot tracks, and it is converted into a GENFIT2 or GENFIT `Track` by means of the *adapters*(2), which are able to deal with changes in the GENFIT2 code without changing additional packages.

In order to test the performance of the GENFIT2 and the new `GenfitTools` package, a so-called particle-gun generator has been used. Five particle hypoteses have been tested: electrons, muons, pions, kaons and protons. A MC particle-gun generator is used to produce single par-
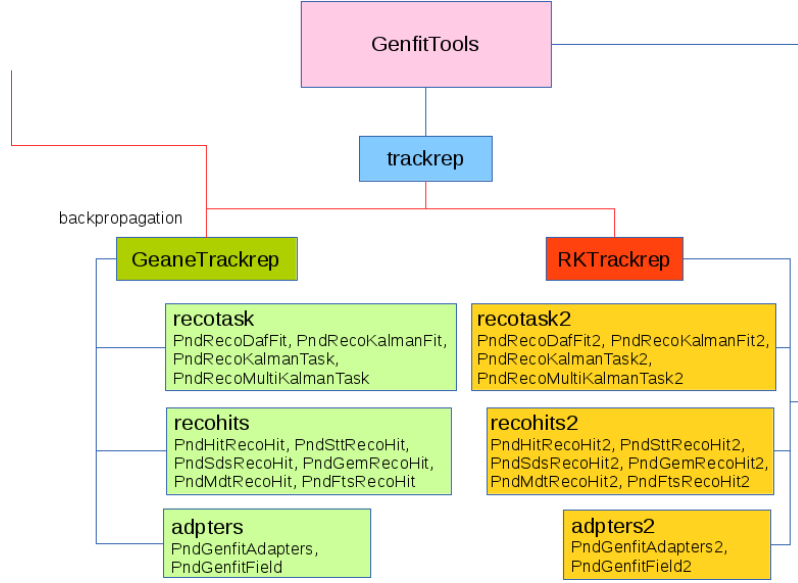
Figure 11: Structure of the `GenfitTools` package, in the PandaRoot trunk 28747 [36].

ticle beam for each different particle hypotheses, with different p_T values, running at the same antiproton beam momentum[5]. For the example mentioned in this section a beam momentum of 15 GeV/c was used. We tested particles emitted at a fixed polar angle ($\theta = 60^0$). We also checked the GENFIT2 performance in PandaRoot by changing the beam momentum, to evaluate the resolution and pull of the tracking parameters. The magnetic field map used in these tests is a realistic one, as shown in Fig. 12(a). Since the $\overline{\mathrm{P}}$ANDA detector is composed by two spectrometers, one with a solenoidal field (on the left part of Fig. 12(a)) and another with dipole field (on the right of Fig. 12(a)), the intermediate region is characterised by strong inhomogeneities which has to be dealt by the extrapolation and fitting code. For comparison, the Belle II magnetic field scheme is shown in Fig. 12(b). More details related to this study are published in Ref. [36].

The track representation used in the following tests is not `RKTrackRep`, because PandaRoot currently uses GEANE [38]; therefore an estimate on how long it takes to reconstruct an event with GENFIT2 in PandaRoot framework is biased, and not comparable with the numbers measured for basf2.

Figure 13 shows the performance of tracking tools in 2 different PandaRoot trunk revisions: the old tool (red squares) in comparison with the new GENFIT2 tool (blue squares). The an-

---

[5]A change in the antiproton momentum beam changes the flux density of the dipole magnetic field in the $\overline{\mathrm{P}}$ANDA forward spectrometer.

tiproton beam momentum is set to 15 GeV/c. No significant differences are observed in the 2 different PandaRoot trunk revisions under investigation, for high p_T; however, for p_T <0.35 GeV/c the GENFIT2 performance is now better. In the PandaRoot trunk revision 28747 a switch is provided to use both, GENFIT2 and GENFIT. GENFIT2 has been introduced in the official PandaRoot releases since 2017.

The result of our simulation shows a clear improvement for reconstruction of low momentum tracks (see Fig. 13). A minimum transverse momentum threshold of 50 MeV/c is imposed, since that is required for at least one hit in the Straw-Tube-Tracker (STT). GENFIT2 is built in a way to track low momentum tracks down 0 MeV/c.

We summarize in Fig. 14 a preliminary achievement of this work in PandaRoot: it shows the pull distribution of the spatial tracking parameters d_0 and z_0, running the particle-gun beam at $p_T = 1$ GeV/c. As expected, the width of a Gaussian fit to the distribution is consistent with 1.0, and the mean with zero. Small variations are observed if we decrease the p_T value, but they are not significant, and the results confirm the stability of the fitter [40].

9   Conclusion

In summary, we have demonstrated that GENFIT2 can successfully provide track-fitting tools when the magnetic field is nearly homeogeneous (such as in Belle II) and when it is non-homogeneous (such as in $\overline{\text{P}}$ANDA). GENFIT2 has shown good track-fitting performance and has been validated with MC samples. It has successfully been integrated into PandaRoot and basf2. It easily adapts to different detector setups and geometries. With the new fitter implementation classes, compared to the former GENFIT2, the performance of low momentum track reconstrucion is improved. Tests have been performed changing particle hypotheses: electron, muon, pion, kaon, and proton. The Runge-Kutta track representation works well in basf2, and is available also in PandaRoot now. The newly available interfaces to Millepede II and RAVE are also useful and easy to utilize, as it is the new event display feature included in the toolkit. Preliminary results are promising, as reported in Ref. [35] (cosmics, and first data collected during the early time of Phase 2 of Belle II for a total integrated luminosity equal to 5 pb^{-1}). GENFIT2 is now used in several projects, including Belle II, COMET, $\overline{\text{P}}$ANDA, FOPI, SHIP, FOOT and BGO-OD. We encourage scientists to make use of our generic tracking tool and participate in our ongoing work to further improve and extend GENFIT2.
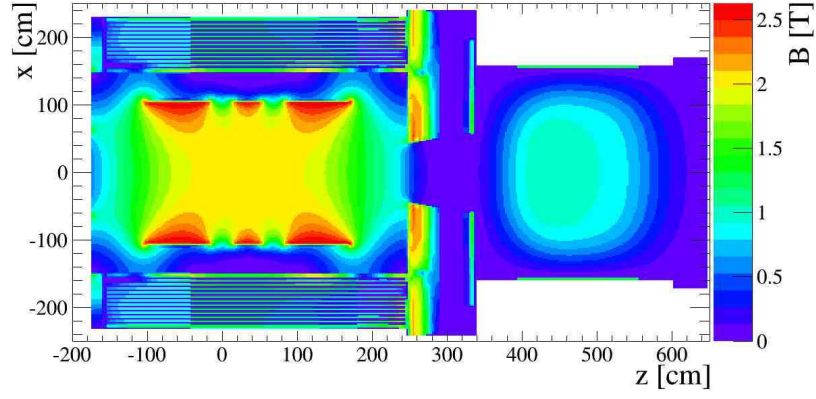
(a)
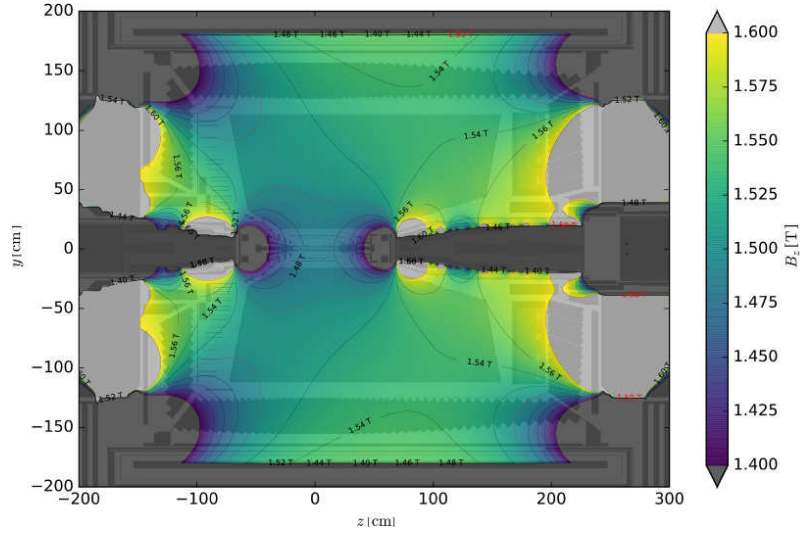
(b)

Figure 12: (a) Magnetic field scheme in $\overline{\mathrm{P}}$ANDA [36]: in the central part of the detector it is constant and equal to 2T; in the dipole area the maximum bending power is 2 T·m. (b) Magnetic field scheme in Belle II [37]: it is a solenoidal field of 1.5 T at the center of the detector.
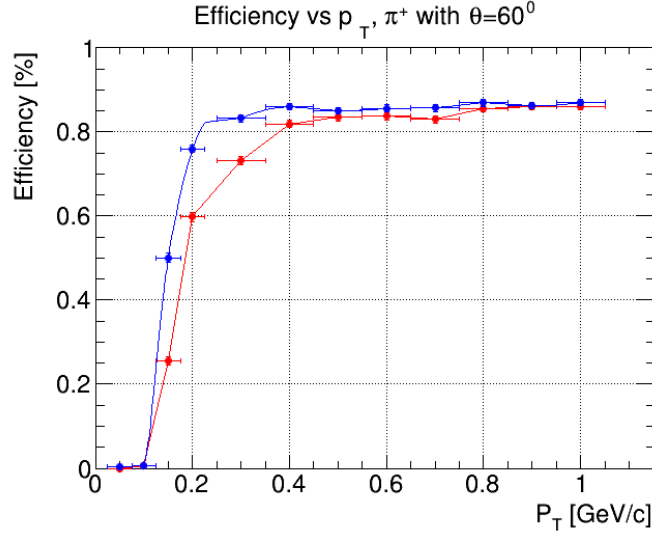
Figure 13: Efficiency versus the transverse momentum (p_T) of charged pions, emitted at the polar angle value $\theta = 60^0$, in PandaRoot. A comparison between the old tool (red squares) and the new GENFIT2 tool (blue points) is provided. The track fitting efficiency is evaluated in a 3σ window around the reconstructed momentum [36].

(a)   (b)

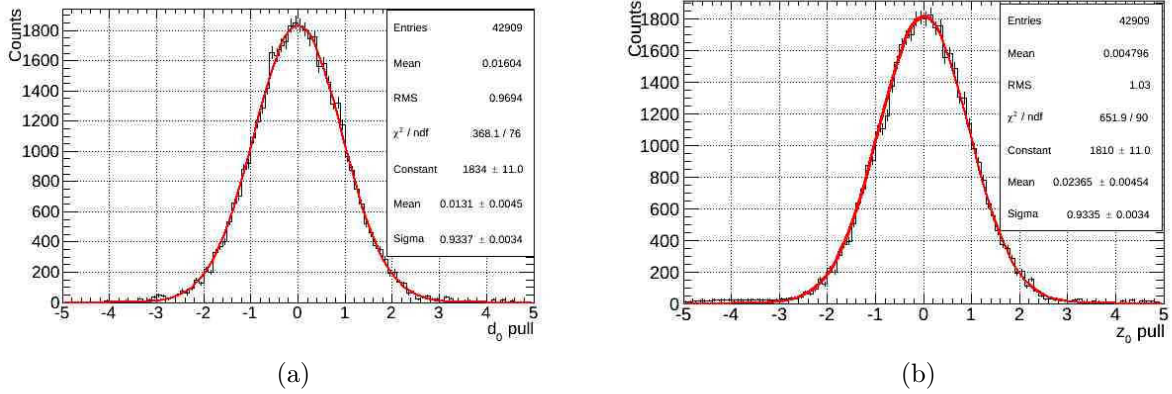Figure 14: Generated values are subtracted from the reconstructed values of the variables d_0 and z_0, and then divided by their errors. These plots have been obtained for $p_T = 400$ MeV/c and they show the pull distributions [36] using GENFIT2 in PandaRoot. The width of the Gaussian fit to both distributions is consistent with being 1.0, as expected for a correct pull distribution, and the mean is consistent with being zero.

10   Bibliography

References

[1] J. Rauch and T. Schlüter, arXiV:1410.3698 [physics.ins-det].

[2] C. Höppner *et al.*, NIM A 620, 518-525 (2010).

[3] M. Al-Turany *et al.*, J. Phys.: Conf. Ser. 119, 032011 (2008).

[4] S. Spataro, J. Phys.: Conf. Ser. 396, 022048 (2012).

[5] W. Erni *et al.*, arXiv:0903.3905 [hep-ex].

[6] T. Bilka *et al.*, IEEE Trans. Nucl. Sci. Vol. 62 Issue 3, 1 (2015).

[7] https://www.belle2.org/

[8] J. Brodzicka *et al.*, Progr. Theor. Exp. Phys. 04D001 (2012).

[9] V. Blobel, NIM A 566, 5 (2006).

[10] T. Abe *et al.*, KEK Report 2010, arXiV:1011.0352 [physics.ins-det].

[11] T. Kuhr *et al.*, arXiv:1809.04299 [physics.comp-ph]

[12] W. Waltenberger. IEEE Trans. Nucl. Sci., 58, 434-444 (2011).

[13] R. E. Kalman, Transcription of ASME - Jou. Basic Eng. 82, D35 (1960).

[14] M. Abramowitz *et al.*, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, Dover, New York (1964).

[15] J. Myrheim *et al.*, NIM 160, 43-48 (1979).

[16] L. Buggie *et al.*, NIM 179, 365-381 (1981).

[17] E. Lund *et al.*, JINST 4, P04001 (2009).

[18] E. Lund *et al.*, JINST 4, P04016 (2009).

[19] GEANT - CERN Program Library Long Write-up, W5013 (1993).

[20] R. Brun and F. Rademakers, AIHENP96 Workshop (Lausanne) 389, 81-86 (1996).

[21] GEANT4 Coll., GEANT 4 NIM A 506, 250-303 (203).

[22] A. Fontana *et al.*, JPCS 117, 032018 (2008).

[23] V.L.Highland NIM 129, 479 (1975); G.R. Lynch and O.I. Dahl NIM B58, 6 (1991).

[24] R. Frühwirth *et al.*, *Data Analysis Techniques for High − Energy Physics*. Cambridge University Press (2000).

[25] B. D.O. Anderson and J. B. Moore, *Optimal Filtering*. Dover Publications (1979).

[26] R. Frühwirth, NIM A 262,444-450 (1987).

[27] R. Frühwirth and A. Strandlie, *Track Fitting with Ambiguities and Noise*: *A Study of Elastic Tracking and Nonlinear Filters*. Computer Physics Communications, 197-214 (1999).

[28] V. Blobel and C. Kleinwort, NIM A 673, 107-110 (2012).

[29] V. Bloblel, NIM A 566 (2006) 14-17.

[30] V. Bloblel and C. Kleinwort, Proc. Conf. ASTPP (2002).

[31] CMS Collaboration, JINST, 0803:S08004 (2008).

[32] J. Rauch and T. Schlüter, JPCS 608, 012042 (2015).

[33] Scheme presented on behalf of the Belle II Coll. by S. Spataro at the CHEP conference, Sophia, 2018.

[34] *The Belle II Physics Book*, arXiV:1808.10567 [hep-ex], pag.60-61.

[35] E. Prencipe, arXiV: 1808.04721 [hep-ex] (submitted to EPJ).

[36] E. Prencipe, JPCS 898, 042037 (2017).

[37] *The Belle II Physics Book*, arXiV:1808.10567 [hep-ex], pag.56.

[38] V. Innocente *et al.*, *GEANE: Average Tracking and Error Propagation Package, CERN*, Program Library W5013-E (1991)

[39] D. N. Brown *et al.*, *The BaBar Track Fitting Algorithm*, Proc. CHEP Conf. (2000).

[40] E. Prencipe, talk given at the *Common Track Reconstruction Software Forum*, Vienna (2016).

[41] G. H. Golub, Num. Math. **7**, 206 (1965).

[42] P. Businger *et al.*, Num. Math. **7**, 269-276 (1965).

[43] B.D.O. Anderson and J.B. Moore, *Optimal filtering*, Dover, 2005, pag. 5.

A   Coordinate system

GENFIT2 uses three different sets of variables to characterize trajectories. Their definitions and the conversions between them is discussed below.

We start by introducing the conventions for labelling planes. A plane in three-space is defined by three vectors: we call its origin $\vec{o}$ (0,0,0), and define the two orthogonal unit vectors u, v, giving the directions of the coordinate axes; the normal of the plane points in the direction of $\vec{n}$ = $\vec{u} \times \vec{v}$. $\vec{v}$, $\vec{u}$ and $\vec{n}$ identify a left-handed system, respectively, as shown in Fig. 2, Sec. 2.

The three sets of variables are:

- *6D-coordinates.* This is the common set used for user interfacing. It contains the cartesian coordinates for the position and the momentum vector, labeled as:

$$(\mathbf{x}, \mathbf{p}).$$

These, together with an independently provided charge q, specify a helix uniquely.

- *local coordinates.* These are the five coordinates used in track fitting. They are given relative to a plane with local orthogonal coordinates u and v. The coordinates are labeled as:

$$(q/p, u', v', u, v)$$

Here q is the charge of the particle; p is its momentum; u', v' are the direction cosines relative to the plane; u, v are the local coordinates on the plane.

These variables are supplemented by a variable $\sigma = \pm 1$, indicating whether the vertical component of the track enters the supporting plane parallel or antiparallel to its normal vector. Conversion to the 6D-coordinates is achieved as follows:

$$\begin{pmatrix} q/p \\ u' \\ v' \\ u \\ v \end{pmatrix} \rightarrow \begin{pmatrix} \mathbf{O} + \mathbf{u}\mathbf{U} + \mathbf{v}\mathbf{V} \\ \frac{\sigma\mathbf{N} + u'\mathbf{U} + v'\mathbf{V}}{\sqrt{1 + u'^2 + v'^2}} \times p \end{pmatrix}$$

- *global coordinate.* These are the seven coordinates used in track extrapolation: the position **x**, the direction unit vector **T** and charge over momentum of the track, written as:

$$(\mathbf{x}, \mathbf{T}, q/p)$$

Conversion to the 6D-coordinates is performed by the mapping:

$$(\mathbf{x}, \mathbf{T}, q/p) \rightarrow (\mathbf{x}, p\mathbf{T}).$$

These coordinates contain the complete helix information. Additionally, we can represent a track by its helix parameters, which are labeled as:

$$(d_0, \phi, \omega, z_0, tan\lambda)$$

and which are defined following the conventions used at BaBar [39].

Jacobians are used for the various possible coordinate transformations. For the mapping between two sets of vectors, $x \rightarrow y$, we define the Jacobian matrix as:

$$J(x \rightarrow y) = (\frac{\partial y}{\partial x}), \tag{19}$$

where the y labels the rows and the x labels the columns. This, together with matrix multiplication, gives the usual chain rule for the mapping $x \rightarrow y \rightarrow z$,

$$J(x \rightarrow z) = J(y \rightarrow z)J(x \rightarrow y) \tag{20}$$

and we obtain the usual error propagation formula:

$$C_y = J(x \rightarrow y) \cdot C_x \cdot J(x \rightarrow y)^T \tag{21}$$

which expresses the covariance matrix of the y in terms of the error matrix of the x together with the Jacobian of the mapping between them.

$$J(local \rightarrow global) = \begin{pmatrix} 0 & 0 & 0 & U_x & V_x \\ 0 & 0 & 0 & U_y & V_y \\ 0 & 0 & 0 & U_z & V_z \\ 0 & \frac{-u'\sigma N_x - u'v'V_x + (1+v'^2)U_x}{(1+u'^2+v'^2)^{3/2}} & \frac{-v'\sigma N_x - u'v'U_x + (1+u'^2)V_x}{(1+u'^2+v'^2)^{3/2}} & 0 & 0 \\ 0 & \frac{-u'\sigma N_y - u'v'V_y + (1+v'^2)U_y}{(1+u'^2+v'^2)^{3/2}} & \frac{-v'\sigma N_y - u'v'U_y + (1+u'^2)V_y}{(1+u'^2+v'^2)^{3/2}} & 0 & 0 \\ 0 & \frac{-u'\sigma N_z - u'v'V_z + (1+v'^2)U_z}{(1+u'^2+v'^2)^{3/2}} & \frac{-v'\sigma N_z - u'v'U_z + (1+u'^2)V_z}{(1+u'^2+v'^2)^{3/2}} & 0 & 0 \end{pmatrix} \tag{22}$$

$$J(6D \rightarrow global) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{p_y^2+p_z^2}{p^{3/2}} & -\frac{p_x p_y}{p^{3/2}} & -\frac{p_x p_z}{p^{3/2}} \\ 0 & 0 & 0 & -\frac{p_x p_y}{p^{3/2}} & \frac{p_x^2+p_z^2}{p^{3/2}} & -\frac{p_y p_z}{p^{3/2}} \\ 0 & 0 & 0 & -\frac{p_x p_z}{p^{3/2}} & -\frac{p_y p_z}{p^{3/2}} & \frac{p_x^2+p_y^2}{p^{3/2}} \\ 0 & 0 & 0 & -\frac{qp_x}{p^{3/2}} & -\frac{qp_y}{p^{3/2}} & -\frac{qp_z}{p^{3/2}} \end{pmatrix} \tag{23}$$

$$J(global \rightarrow 6D) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p & 0 & 0 & -p^2 q T_x \\ 0 & 0 & 0 & 0 & p & 0 & -p^2 q T_y \\ 0 & 0 & 0 & 0 & 0 & p & -p^2 q T_z \end{pmatrix} \tag{24}$$

The important special cases, where the plane is orthogonal to the track, *i.e.* $u' = v' = 0$, delivers:

$$J(local \rightarrow 6D) = J(global \rightarrow 6D) \cdot J(local \rightarrow global) = \begin{pmatrix} 0 & 0 & 0 & U_x & V_x \\ 0 & 0 & 0 & U_y & V_y \\ 0 & 0 & 0 & U_z & V_z \\ -p^2 q \sigma N_x & U_x & V_x & 0 & 0 \\ -p^2 q \sigma N_y & U_y & V_y & 0 & 0 \\ -p^2 q \sigma N_z & U_z & V_z & 0 & 0 \end{pmatrix} \tag{25}$$

In order to express efficiently the Jacobian for conversion from 6D to local coordinates, we express the momentum vector in its components (p_U, p_V, p_N), where, *e.g.* $p_U \equiv \mathbf{p} \cdot \mathbf{U}$, and $\sigma = \mathrm{sgn}(\mathbf{p_N})$. Then:

$$J(6D \rightarrow local) = \begin{pmatrix} 0 & 0 & 0 & -\frac{q(p_N N_x + p_U U_x + p_V V_x)}{p^3} & -\frac{q(p_N N_y + p_U U_y + p_V V_y)}{p^3} & -\frac{q(p_N N_z + p_U U_z + p_V V_z)}{p^3} \\ 0 & 0 & 0 & \frac{\sigma(p_N U_x - p_U N_x)}{p_N^2} & \frac{\sigma(p_N U_y - p_U N_y)}{p_N^2} & \frac{p_N U_z - p_U N_z}{p_N^2} \\ 0 & 0 & 0 & \frac{\sigma(p_N V_x - p_v N_x}{p_N^2} & \frac{\sigma(p_N V_y - p_V N_y)}{p_N^2} & \frac{p_N V_z - p_V N_z}{p_N^2} \\ U_x & U_y & U_z & 0 & 0 & 0 \\ V_x & V_y & V_z & 0 & 0 & 0 \end{pmatrix} \tag{26}$$

$$J(global \rightarrow local) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{(\mathbf{T \cdot N})U_x - (\mathbf{T \cdot N})N_x}{(\mathbf{T \cdot N})^2}\sigma & \frac{(\mathbf{T \cdot N})U_y - (\mathbf{T \cdot N})N_y}{(\mathbf{T \cdot N})^2}\sigma & \frac{(\mathbf{T \cdot N})U_z - (\mathbf{T \cdot N})N_z}{(\mathbf{T \cdot N})^2}\sigma & 0 \\ 0 & 0 & 0 & \frac{(\mathbf{T \cdot N})V_x - (\mathbf{T \cdot N})N_x}{(\mathbf{T \cdot N})^2}\sigma & \frac{(\mathbf{T \cdot N})V_y - (\mathbf{T \cdot N})N_y}{(\mathbf{T \cdot N})^2}\sigma & \frac{(\mathbf{T \cdot N})V_z - (\mathbf{T \cdot N})N_z}{(\mathbf{T \cdot N})^2}\sigma & 0 \\ U_x & U_y & U_z & 0 & 0 & 0 & 0 \\ V_x & V_y & V_z & 0 & 0 & 0 & 0 \end{pmatrix} \tag{27}$$

Again, the special case where momentum is orthogonal to the plane in question $(\mathbf{T} \cdot \mathbf{N} = \frac{1}{p}\mathbf{p} \cdot \mathbf{N}$

$= \sigma = \pm 1$), gives:

$$
J(6D \to local) =
\begin{pmatrix}
0 & 0 & 0 & -\frac{q}{p^2}N_x & -\frac{q}{p^2}N_y & -\frac{q}{p^2}N_z \\
0 & 0 & 0 & \sigma\frac{U_x}{p} & \sigma\frac{U_y}{p} & \sigma\frac{U_z}{p} \\
0 & 0 & 0 & \sigma\frac{V_x}{p} & \sigma\frac{V_y}{p} & \sigma\frac{V_z}{p} \\
U_x & U_y & U_z & 0 & 0 & 0 \\
V_x & V_y & V_z & 0 & 0 & 0
\end{pmatrix}
\tag{28}
$$

and:

$$
J(global \to local) =
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & \sigma U_x & \sigma U_y & \sigma U_z & 0 \\
0 & 0 & 0 & \sigma V_x & \sigma V_y & \sigma V_z & 0 \\
U_x & U_y & U_z & 0 & 00 & 0 & 0 \\
V_x & V_y & V_z & 0 & 00 & 0 & 0
\end{pmatrix}
\tag{29}
$$

As consistency check, one can try to calculate the transformation: $local \to global \to local$: the Jacobian of this transformation is indeed equal to 0; the converse transformation, *e.g.* $global \to local \to global$, has singular Jacobian. Its null space is spanned by: $(0, 0, 0, \mathbf{T_x}, \mathbf{T_y}, \mathbf{T_z})$, $(\mathbf{N_x}, \mathbf{N_y}, \mathbf{N_z}, 0, 0, 0, 0)$, which comes about because T is normalized (this brings about the first direction of the null space) and because a point in local coordinates is constrained to a plane (the second direction).

B   Averaging

Given two measurements of the same n-dimensional quantity and their respective covariances, (x_1, C_1) and (x_2, C_2), the two measurements can be combined to a least-likelihood estimate $\tilde{x}$ by minimizing the χ^2 of the combination, *i.e.*

$$
\chi^2 = (\tilde{x} - x_1)^T C_1^{-1}(\tilde{x} - x_1) + (\tilde{x} - x_2)^T C_2^{-1}(\tilde{x} - x_2) \equiv \min_{\tilde{x}}.
\tag{30}
$$

The solution is the well-known formula:

$$
\tilde{x} = (C_1^{-1} + C_2^{-1})^{-1}(C_1^{-1}x_1 + C_2^{-1}x_2),
\tag{31}
$$

and the covariance of $\tilde{x}$ is by linear error propagation:

$$
\tilde{C} = (C_1^{-1} + C_2^{-1})^{-1}.
\tag{32}
$$

We can simplify the evaluation of these textbook formulas significantly in the following manner, applying a technique from Ref. [41]: covariance matrices are positive definite matrices. As such,

they can be written $C = S^T S$ where S is some matrix which can be chosen to be upper triangular. In the case where C is diagonal, the entries of C correspond to the squared errors σ^2 and the entries of S correspond to the errors σ. It is in this sense that S is referred to as a square root of C. It is commonly evaluated by Cholesky decomposition. Since S is a upper triangular matrix its inverse, and thus the inverse of C, can be evaluated directly by means of forward substitution. Likewise products of the form $S^{-1}x$ can be evaluated by forward substitution without first evaluating the inverse. Our goal is to replace the matrices in Eq. (31) and Eq.(32) by their square roots and use those directly, thus operating with the errors instead of the squared errors.

Starting with the inner part of (32), we can write:

$$C_1^{-1} + C_2^{-1} = S_1^{-1T}S_1^{-1} + S_2^{-1T}S_2^{-1} \tag{33}$$

$$= (S_1^{-1T}, S_2^{-1T}) \begin{pmatrix} S_1^{-1} \\ S_2^{-1} \end{pmatrix}, \tag{34}$$

where on the second line we use block matrix notation, *i.e.* the matrix on the right is $2n \times n$. As such, there is an $2n \times 2n$-dimensional orthogonal matrix Q, $QQ^T = 1$, which converts it to an upper triagonal matrix, written explicitly:

$$\begin{pmatrix} S_1^{-1} \\ S_2^{-1} \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{35}$$

where R is $n \times n$. This operation is the QR decomposition of the left-hand side, and we can write:

$$C_1^{-1} + C_2^{-1} = (R^T, 0)QQ^T \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{36}$$

$$= R^T R. \tag{37}$$

Going back to Eq. (32), we have:

$$\tilde{C} = R^{-1}R^{-1T}. \tag{38}$$

In other words, we can obtain the square root of $\tilde{C}$ directly from the square roots of C_1, C_2 by stacking them and performing a QR decomposition. This increases the numerical range in the presence of vastly different errors, as we no longer operate on the squared errors but on the errors themselves. Additionally, any matrix of the form in Eq. (38) is always invertible and positive definite which prevents catastrophic conditions in subsequent numerical operations.

It is perhaps revealing to consider the similarity to the Pythagorean theorem which gives the length of a vector $(a, b)^T$ as $a^2 + b^2 = c^2$. The corresponding orthogonal transformations in this case are the rotations which take the vector $(a, b)^T$ to $(\pm\sqrt{a^2 + b^2}, 0)^T$. The computational gain comes from the fact that the orthogonal operation disappears because the relevant quantities are squares which always lead to factors QQ^T or Q^TQ (for the inverse matrices).

In order to evaluate $\tilde{x}$, we insert those into Eq. (31), and obtain:

$$\tilde{x} = R^{-1}R^{-1T}(R^T, 0)Q \begin{pmatrix} S_1^{-1}x_1 \\ S_2^{-1}x_2 \end{pmatrix} \tag{39}$$

$$= R^{-1}\bar{Q} \begin{pmatrix} S_1^{-1}x_1 \\ S_2^{-1}x_2 \end{pmatrix}, \tag{40}$$

where $\bar{Q}$ refers to the upper n rows of Q. It never explicitly appears in the implementation, instead the right-hand multiplication is evaluated simultaneously with the QR decomposition in Eq. (35), using the algorithm of Ref. [42].

In practice this algorithm is not only preferable because of its numerical properties, we also measured it to perform slightly faster than a naive implementation of Eqs. (31) and (32).

C   Proof of Square-root Kalman filter formula

In a similar vein to Appendix B, we want to express the expressions for $\boldsymbol{p}_{i|i}$ and the square root of $C_{i|i} = \tilde{S}^T\tilde{S}$ in Eq.(15) in terms of the $n \times n$ square roots of $C_{i|i-1} = S^TS$ and the $m \times m$ measurement covariance $V_i = R^TR$. Following Ref. [43], we want to show that we can find corresponding expressions by QR decomposition of the matrix:

$$\begin{pmatrix} R & 0 \\ SH^T & S \end{pmatrix} = Q \begin{pmatrix} U & X \\ 0 & W \end{pmatrix}, \tag{41}$$

where $QQ^T = 1$, U and W are upper triagonal matrices, and the block sizes on the right-hand and left-hand sides are equal. For sake of brevity we write $H = H_i$, $V = V_i$, $C = C_{i|i-1}$.

We want to show that $W = \tilde{S}$ and $X^TU^{-1T} = CH^T(V + HCH^T)^{-1}$. In order to do this, we square both sides of (41), *i.e.* multiply each side with its transposed from the left. This gives:

$$\begin{pmatrix} V + HCH^T & HC \\ CH^T & C \end{pmatrix} = \begin{pmatrix} U^TU & U^TX \\ X^TU & X^TX + W^TW \end{pmatrix}. \tag{42}$$

From this we can read off $U^TU = V + HCH^T$, $U^TX = HC$. We can now find X^TX by strategical insertion of the $m \times m$ identity matrix followed by a rearrangement of parentheses,

$$X^TX = X^T \cdot 1 \cdot X = X^T(UU^{-1})(U^{-1T}U^T)X = (X^TU)(U^TU)^{-1}(U^TX). \tag{43}$$

By identifying the corresponding entries in (42), we see that

$$X^T X = CH^T (V + HCH^T)^{-1} HC. \tag{44}$$

Inserting this into the lower right block of (42) and comparing to (15), we find that:

$$W^T W = C - CH^T (V + HCH^T)^{-1} HC = \tilde{S}^T \tilde{S} \tag{45}$$

and thus identify $W = \tilde{S}$, the desired square root of the updated covariance matrix. Finally, by another strategic insertion of 1 we find:

$$X^T U^{-1^T} r_i = X^T U (U^T U)^{-1} r_i = CH^T (V + HCH^T)^{-1} r_i, \tag{46}$$

i.e. the update of the state. Observe how the more familiar formulas are obtained by inserting additional, cancelling terms in Eqs. (43) and (46). These cancellations are what degrade the numerical behavior of the more commonly used formulas.

D   List of acronyms

BASF = Belle Analysis Software Framework

CDC = Central Drift Chamber

CKF = Combined Kalman Filter

DAF = Deterministic Annealing Filter

EM = Electro-Magnetic calorimeter

FAIR = Facility for Antiproton and Ion Research

GENFIT = General Fitter

GSI = Gesellschaft für Schwerionenforschung (Helmholtz Centre for Heavy Ion Research)

GBL = Global Broken Line

POCA = Point of Closest Approach

PXD = Pixel Detector

RAVE = Reconstruction of vertices in Abstract Versatile Environments

SVD = Silicon Vertex Detector

STT = Straw-Tube Tracker

TOP = Time-Of-Propagation detector