The ATLAS Track Extrapolation Package

A. Salzburger*

Leopold Franzens Universität Innsbruck, Austria & CERN

December 11, 2007

Abstract

The extrapolation of track parameters and their associated covariances to destination surfaces of different types is a very frequent process in the event reconstruction of high energy physics experiments. This is amongst other reasons due to the fact that most track and vertex fitting techniques are based on the first and second momentum of the underlying probability density distribution. The correct stochastic or deterministic treatment of interactions with the traversed detector material is hereby crucial for high quality track reconstruction throughout the entire momentum range of final state particles that are produced in high energy physics collision experiments. This document presents the main concepts, the algorithms and the implementation of the newly developed, powerful ATLAS track extrapolation engine. It also emphasises on validation procedures, timing measurements and the integration into the ATLAS offline reconstruction software.

*E-mail: Andreas.Salzburger@cern.ch

1 Introduction

The transport of track parameters (i.e. the representation of a track with respect to a given surface) and their associated covariances is a very frequent process in track reconstruction. Most progressive fitting techniques such as the Kalman filter formalism [1] rely on the prediction of the gathered track information on the successive measurement surface (a simplified illustration of a track extrapolation in a typical Kalman filter step can be seen in Fig. 1). In global fitting techniques, on the other hand, the prediction of the track depending on the initial parameters (i.e. the fitted parameters) enters the global χ^2 function to be minimized. For both, global and sequential track fitting algorithms, the correct treatment of effects caused by the interaction of the particle with traversed detector material is essential; in the least squares fit the uncertainties due to material interactions regulate the contribution of the fitted scattering angle to the global χ^2 function. In most sequential fitting techniques the uncertainties of the momentum direction and magnitude caused by interactions with the detector material are directly applied as deterministic energy loss and additional contributions to the covariance matrix during the extrapolation process.

Track extrapolation is furthermore necessary in vertex fitting, where the expression of the track with respect to the estimated vertex position has to be evaluated iteratively towards convergence. Moreover, in pattern recognition the seeded prediction of a track may be used for trajectory building and hit finding. Finally, the track parameters representation on a destination surface is needed for combined reconstruction to enhance the matching of tracking information and calorimeter clusters on the one hand, and the combination of tracks and track segments from different tracking devices on the other hand.

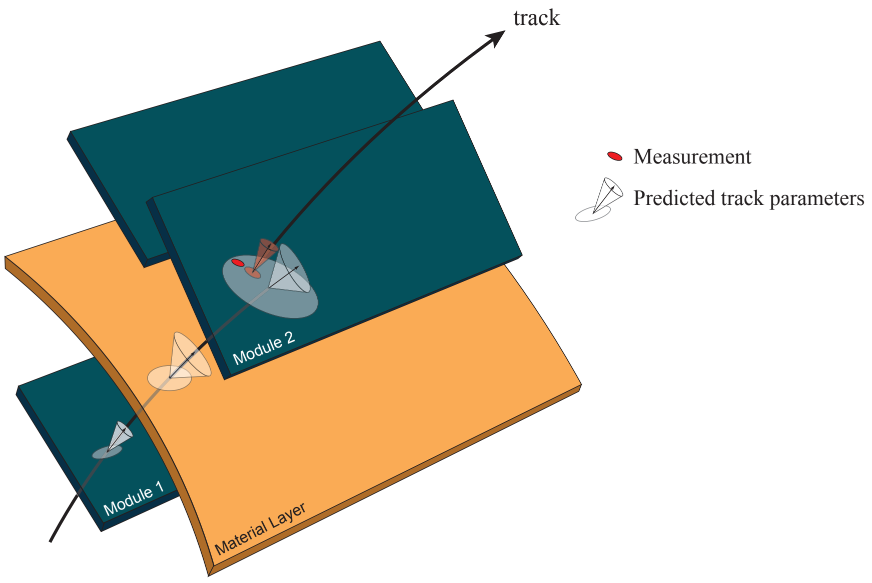


Figure 1: Simplified illustration of a typical extrapolation process within a Kalman filter step. The track representation on the detector module 1 is propagated onto the next measurement surface, which results in the track prediction on module 2. The traversing of the material layer between the two modules causes an increase of the track direction uncertainties and thus — by correlation — an increased uncertainty of the predicted track parameters. In the Kalman filter formalism, the weighted mean between prediction and associated measurement build the updated measurement which builds the start point for the next filter step; this leads to the illustrated non-continuous track model.

The ATLAS experiment puts stringent requirements to the track reconstruction software: for the track parameter propagation, in particular, the highly inhomogeneous magnetic field setup — a solenoidal field with a central magnitude of 2 Tesla in the *Inner Detector* (ID) and a toroidal magnetic field with a peak strength higher than 4 Tesla in the *Muon Spectrometer* (MS) — and the complex material distribution of the sub-detectors must be dealt with correctly to achieve a satisfactory tracking resolution. Additionally, the expected high track occupancy in the Inner Detector and the resulting high number of track candidates impose combinatorial and timing constraints for track finding and fitting algorithms. Since the CPU time consumption of the track reconstruction chain has to be minimised to fit with the computing budget of the experiment, the algorithmic solution of a powerful but fast track extrapolation engine is a challenging task.

During the redesign of the ATLAS offline reconstruction software that has been invoked by the *Final Report of the Reconstruction Task Force* (RTF) [2], a new extrapolation package has been designed and developed. First big scale tests of the new extrapolation engine have been performed during event reconstruction of the ATLAS *Combined Test Beam* 2004 (CTB2004), the first commissioning runs using cosmic rays since 2005 and the reconstruction of Monte Carlo simulated data. The extrapolation package is fully integrated in the object oriented C++ based ATLAS software framework ATHENA [3] and has been developed respecting ATLAS coding standards [4]. A description of the main components of the ATHENA framework, the `Service`, the `Algorithm` and the `AlgTool` interfaces, can be found in [5]. This document is based on the ATLAS software release 13.0.10, while extensions that have been applied after this release are indicated within the context.

1.1 Design Principles and Document Structure

The transportation of a track representation to a destination surface can be divided into three conceptionally different tasks that have been independently realised as single components of the ATLAS extrapolation package. For convenience, they will also build the guideline of this document:

- the **propagation** process describes the mathematical transport of the track parameters and associated covariances to the target surface. The propagation module is defined through a dedicated `IPropagator` interface, that is implemented through various different propagation `AlgTool` classes. A more detailed description of the propagation process can be found in Sec. 2.

- the **navigation** relates the trajectory to the various entities of the reconstruction geometry. This is necessary to find the appropriate material description, to ensure the access to the (configured) magnetic field map, or even to find the detector volume that contains the destination surface. A dedicated `INavigator` interface defines these navigation methods, see Sec. 3.

- the **integration of material effects** according to the traversed detector material that is provided by the navigation marks the third column of the extrapolation package. It is enhanced through various different `AlgTool` classes and further described in Sec. 4.

The complete process including all three tasks will in the following be referred to as *track extrapolation*. It is concentrated in a single `AlgTool` implementation, the `Extrapolator`; a schematic illustration of the three column structure that is incorporated by the extrapolation engine — represented through the various interface definitions — can be seen in Fig. 2. A further description of the user interface and the successive steering of propagation, navigation and material effects integration will be described in Sec. 5. Section 6 will give performance numbers in both timing and accuracy for typical track reconstruction applications. The Appendix explains used conventions and typesetting formats, and gives an exhaustive list of formulas used within the various track propagation techniques.

1.2 Component Pattern and Data Factory Design

The TrkExtrapolation repository follows a strict component pattern design, i.e. abstract interface classes are concentrated in dedicated interface packages, while concrete implementations of algorithmic code and data classes are separated in different component and installed libraries, respectively. This allows dynamic loading of libraries at job execution level, and is thus the key to the high flexibility. The single components of the track extrapolation package are, in general, realised as factories, i.e. the performed operations lead to newly created objects. In C++ terms this is done by returning pointers to objects that are dynamically created using the `new` operator, while the return of a pointer with value 0 indicates that the operation could not be performed. The ATLAS tracking data model, briefly presented in the following section, does not foresee invalid objects, such as e.g. a representation of a failed extrapolation process.

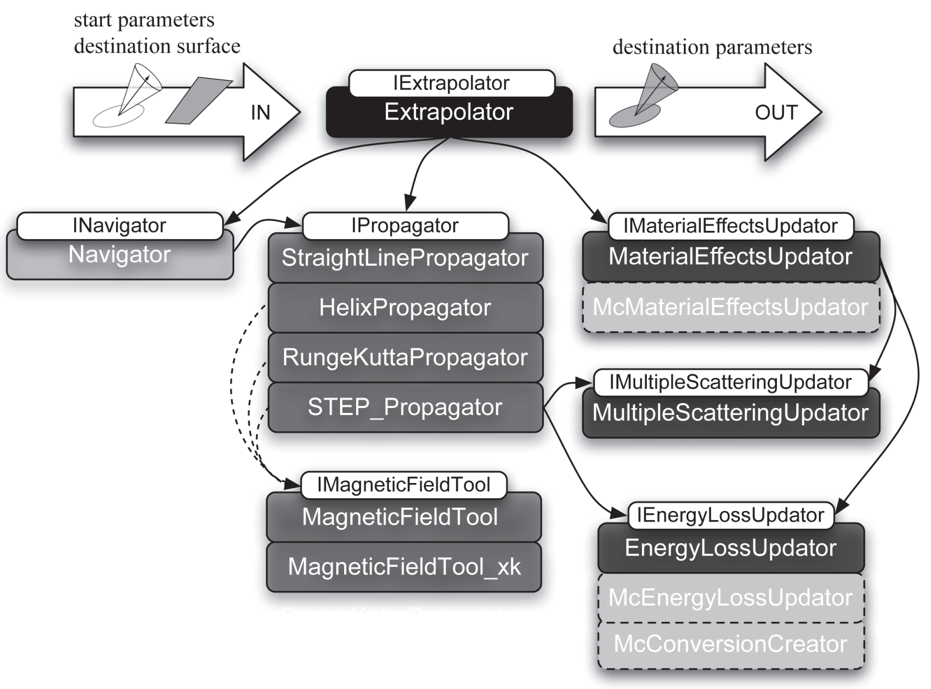


Figure 2: The ATLAS extrapolation engine illustrated as a task diagram. The various modules of propagation, navigation and material effects integration are identified through dedicated interfaces. The main concrete implementations are also shown in this diagram, brighter colored boxes framed by a dashed line represent hereby the exchanged modules for the use of the extrapolation engine in the fast track simulation application.

1.3 Tracking Event Data Model and `TrackingGeometry`

The extrapolation package is based on the common tracking event data model (EDM) and the ATLAS reconstruction geometry. In particular, the main EDM classes extending the `TrackParameters` base class and the geometry classes `Surface`, `TrackingGeometry`, `TrackingVolume` and `Layer` are essential ingredients of the track extrapolation process. The full description of these classes would go far beyond the scope of this document the interested reader is, however, encouraged to read further in [6] and [7], respectively, for an exhaustive description of these modules.

Track Parameterisation A track can be parameterised with respect to a surface in many different ways. If a particle propagates through magnetic field, however, a minimal set of five parameters is necessary to provide a complete and unambiguous parameterisation with respect to any given detector surface. In the following, such a representation of the trajectory will be referred to as `TrackParameters`. In ATLAS, dedicated `TrackParameters` objects exist for every defined surface type and provide the following track parameterisation[1]

$$\mathbf{x} = (l_1, l_2, \phi, \theta, q/p)^T \tag{1}$$

when l_1 and l_2 denote the local coordinates on the given surface (and thus depend on the surface type), ϕ and θ are the azimuth and polar angle, respectively, and q/p is the inverse momentum multiplied by the charge q. For the widely used perigee representation that is often used to express the track with respect to the nominal beam axis the local parameters l_1 and l_2 correspond to the transverse and longitudinal impact parameter d_0 and z_0, respectively. Figure 3 shows an illustration of the perigee representation using ATLAS conventions.

[1]Many experiments in the past used a — for many analysis aspects — more handy parameterisation of a helix with a curvature-optimised momentum representation q/p_T and the polar direction represented as $\cot\theta$. Since for ATLAS a general representation had to be chosen that is valid throughout the detector and thus within different magnetic field setups, a helical representation that is bound to one magnetic frame is not optimal, as the transverse momentum does not remain a constant of motion.

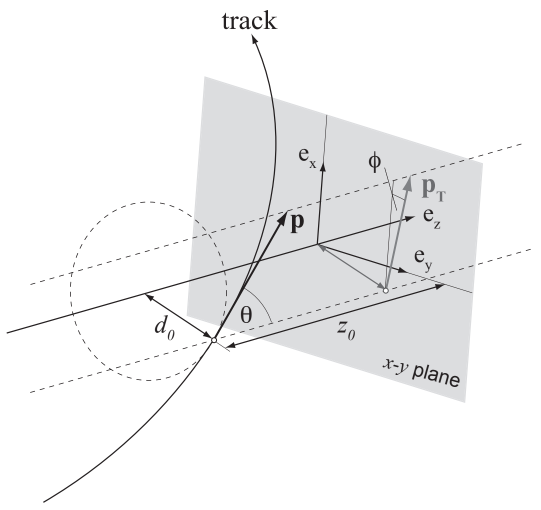


Figure 3: The perigee representation expressed in the ATLAS track parameterisation. The local expression of the point of closest approach is given by the signed transverse impact parameter d_0 and the longitudinal impact parameter z_0. The momentum direction is expressed in global coordinates using the azimuthal angle ϕ that is defined in the projected $x - y$ plane and the polar angle θ, which is measured with respect to the global z axis.

Neutral Parameters Recently, the ATLAS tracking EDM has been extended to deploy a dedicated schema for neutral particle representations [8]. The fifth parameter of the representation as given in Eq. (1) is hereby modified to represent $1/q$, omitting the charge definition. Charged and neutral trajectory representations are realised through the same templated class objects to avoid code duplication, while keeping the type diversity to prevent misinterpretations to happen during the reconstruction flow. The extrapolation package and propagation tools have been adapted to cope with both charged and neutral types, but the ATLAS `Track` class remains restricted to charged trajectories[2]. Neutral parameters are only transported along a straight line to the provided target surface. Material effects are not taken into account and thus the navigation process is not necessary in this context. This documents concentrates therefore on the extrapolation process of charged track representations and will only briefly mention the particularities for neutral parameterisation in the various different modules.

2 Propagation

The mathematical propagation of track parameters to a destination surface is — when omitting energy loss and multiple scattering effects — determined by the starting parameters and the traversed magnetic field. A homogenous magnetic field setup (no field or constant field value and direction) allows to use an underlying parametric track model for the propagation. Many propagation processes can then be solved purely analytically to find the intersection of the track with the destination surface and even for the transported covariances. However, the highly inhomogeneous magnetic field of the ATLAS detector setup requires tracking of particles by numerical methods. Figure 4 shows the magnetic field of the ATLAS detector in an $r - z$ projection for both, the Inner Detector in detail, and the Muon Spectrometer.

The variety of the different propagation techniques is enhanced by different implementations of a common abstract `AlgTool` interface, the `IPropagator`. The interface for propagator `AlgTool` classes is kept very simple; it reflects the pure principle of the task: an input `TrackParameters` object, a destination surface, magnetic field properties and a boolean for the surface bound handling is passed through the method signature, while on the other hand the propagated parameters are returned as the method value. Returning a pointer to a new object puts the responsibility of memory cleanup onto the client algorithm, but complies fully with the factory pattern design described in Sec. 1.2.

The following main interface methods are defined for the `IPropagator` interface:

- The `propagate()` method shall be used in cases when the track parameters to be transported are likely to carry a covariance matrix and the client algorithm relies on the transported error description as well. If the input parameters do not have associated errors, only the parameters are transported to the destination surface.
- To save CPU time, the `propagateParameters()` that only performs the transport of the pa-

[2]This is because neutral particles are not subject of tracking in the classical terms of track finding and track fitting.

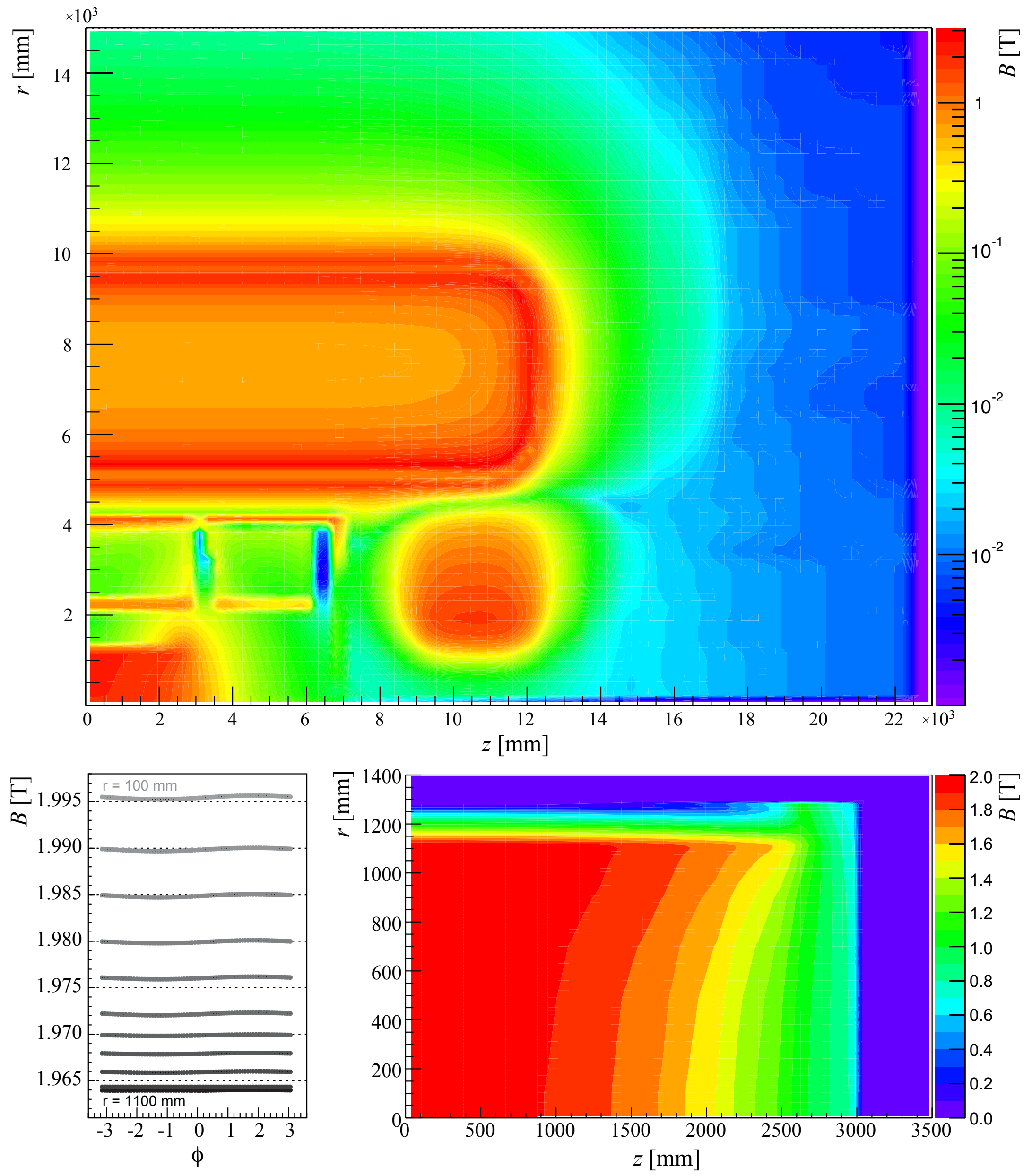


Figure 4: The realistic magnetic field in the $r - z$ plane for the entire ATLAS detector. The upper plot shows the magnetic field strength in the $r - z$ plane at an azimuthal angle of $\phi = \pi/8$ which lies within one Muon System toroid structure. The plots at the bottom focusses on the magnetic field of the Inner Detector as described by the ATLAS-CSC-01-02-00 layout. The first plot at the bottom shows the ϕ-dependency of the magnetic field at different radii in steps of 100 millimeter at $z = 0$: the homogeneity of the field in the ID is broken in radial and azimuthal direction even in the very central part of the solenoid. The second plot shows the magnitude of the magnetic field shown within a quarter of the Inner Detector.

rameters and omits the transport of the associated covariances can be chosen. This is optimised for situations where the transported error represented at the destination surface is not needed.

- The `globalPositions()` method is designed to fill a list with 3D points along the track in intervals of a given step length and confined within a given volume. It is mainly performed during the road building process of the pattern recognition stage.

- The `validationAction()` enables to call event- or track-based validation directives from outside (e.g. such as parameter resetting or the filling of validation information into appropriate output containers). This interface method is not particular to the `IPropagator` interface. In fact, most of the used `AlgTool` classes in the `TrkExtrapolation` realm implement a method of this type and purpose.

Four different propagators have been implemented and are located in different packages of the ATLAS software repository, the `StraightLinePropagator` and `HelixPropagator` incorporate a straight line and respectively helical track model, whereas the `RungeKuttaPropagator` and `STEP_Propagator` are based on the numerical Runge-Kutta-Nystrøm [9] integration technique to evaluate the field integral.

2.1 Intersection with Surfaces and Direction Instructions

The intersection of a track with a surface can be expressed in both global and local coordinates. In terms of track parameters propagation also the direction of the track at the destination surface is needed for a complete track parameterisation: the two local coordinates of the surface intersection and the momentum of the track at the destination surface build the five parameters of the propagated `TrackParameters` expression. The local intersection point — together with the surface constraint — establish the global representation of the trajectory intersection with the target surface; a detailed discussion of the `TrackParameters` object and the used local frame definitions can be found in [6].

The propagation can be performed with a direction instruction, steered by an `enumeration` object; propagations `alongMomentum`, `oppositeMomentum` and `anyDirection` can be performed. The direction does not only determine the intersection solution to be chosen — in case that multiple solutions exist — but also handles the material effects integration during the extrapolation process. For planar surfaces, i.e. the `PlaneSurface` and `DiscSurface` in the ATLAS reconstruction geometry model, there is, in general, one unique solution[3] for the intersection. The *intersection* with a straight line (in the ATLAS reconstruction geometry described by either the `StraightLineSurface` or the `PerigeeSurface`) is defined as the transverse closest approach to the line. The local expression of this point of closest approach is done by a signed transverse impact parameter and the longitudinal impact parameter with respect to the line frame. For surfaces of type `CylinderSurface` typically one or two intersections exist for a given start parameter set. If the propagation direction excludes one of the solutions, obviously the other solution is taken, if both intersections are compatible with the provided directive (or the directive has been chosen to be `anyDirection`), the closer intersection is taken by convention.

2.1.1 The Measurement Frame

The transport of the covariance matrix to a given surface requires the definition of the so-called measurement frame, i.e. the coordinate system attached to the destination surface in which the error on the local parameters is properly defined. For planar surfaces this definition is trivial: the measurement frame is hereby identical with the local surface frame, a cartesian frame for the `PlaneSurface` and a polar frame for the `DiscSurface`. For cylindrical surfaces, the measurement frame is defined — by convention — as the tangential plane to the intersection point with the cylinder. Since in ATLAS no measurement is given on a cylindrical surface (these are mostly needed for the navigation through the volumes of the reconstruction geometry), this choice saves unnecessary conversions from cartesian to cylindrical coordinates (and *vice versa*).

For track expressions with respect to a line (`Perigee` or `AtaStraightLine` definitions) the measurement frame can only be constructed once the point of closest approach is found: it is characterised by the transverse *drift* direction $\mathbf{d}_D$ as the first local axis and the line direction $\mathbf{d}_L$ as the second, perpendicular axis. Figure 5 shows an illustration of the measurement frame for a closest approach to a straight line.

2.2 Analytical Track Models: `StraightLinePropagator` and `HelixPropagator`

Propagations along a straight line can be solved analytically for all surface types. This transport type is applied for particles of zero charge or in absence of a magnetic field. Furthermore, they may

[3]In the ATLAS experiment, the geometrical dimensions of the destination surfaces are in general small compared to the curvature of the track due to the bending of the magnetic field. Hence, multiple intersections with planar surfaces are very rare.

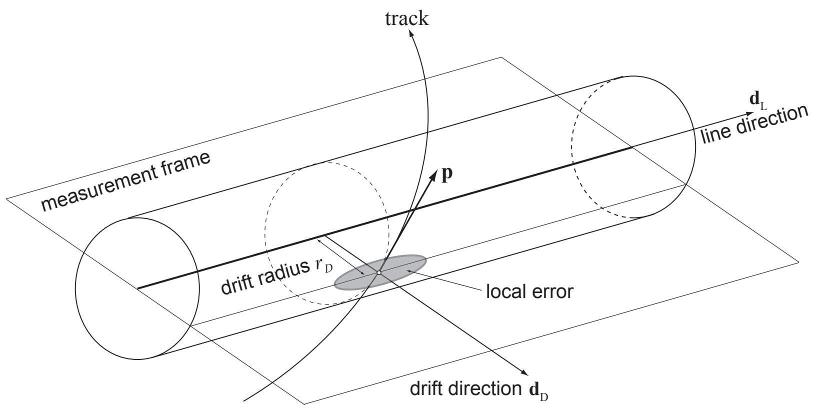


Figure 5: The measurement frame for the closest approach of a track to a nominal line surface. The error on the local coordinates is given in a two-dimensional cartesian system, defined through the drift direction $\mathbf{d}_D$ and the line direction $\mathbf{d}_L$, respectively.

be used for crude estimates, in particular in the non-bending plane. The `StraightLinePropagator`, located in the TrkExSlPropagator package, provides this functionality. It has been extensively used during the reconstruction of events taken during the ATLAS CTB2004 and data taken from the first commissioning runs using cosmic rays, where several periods of data taking without magnetic field have taken place.

The `HelixPropagator` is suitable for propagations of parameters in a homogeneous magnetic field, a very rare or even non existing situation in the ATLAS reconstruction. However, for the CTB2004 Monte Carlo simulation has been done assuming a constant magnetic field and for the new *Fast ATLAS Track Simulation* (FATRAS) [10], the constant magnetic field together with the `HelixPropagator` may be a sufficient simplification for various applications. When using a helical track model the analytical solution for track intersections with many surfaces can only be performed if the surfaces are *aligned* with the guiding center of the helix. Assuming perfect alignment of the ATLAS detector, many surfaces fulfill this requirement. For arbitrarily oriented surfaces, a numerical method to solve the equation has to be used, which is solved through a standard Newton-Rhapson approach. Due to its rare usage in the reconstruction this method has been, however, poorly validated and clearly lacks both accuracy and stability for production usage.

The formulas used for the calculations of the surface intersections for both the `StraightLinePropagator` and the `HelixPropagator` can be found in the Appendix, Sec. A.3 and Sec. A.4.

2.2.1 Analytical Error Propagation along the curvilinear Frame

Since both, the straight line and the helical propagation incorporate an underlying analytical track model, the transport of the covariance matrix can be performed analytically as well. This is done using the so-called *curvilinear* frame that can be defined at every point of the track and has been widely used in the past in high energy physics experiments [11]. The intersection solution with a target surface is, in general, a non-linear function of the starting parameters and it is thus not obvious how to propagate the associated error. A solution to this is to linearise the problem by taking the first order of a Taylor series expansion for the error propagation. The derivatives of the transported parameters with respect to the start parameters build hereby the Jacobian matrix that yield the transformation of the covariance matrix. Within the curvilinear frame, the transport Jacobian matrix $\mathbf{J}_{\mu,\nu}$ becomes an analytical expression with the single parameter s, which denotes the propagation length along the line or helix. Given a global point $\mathbf{m}$ and the momentum $\mathbf{p}$ (defined at $\mathbf{m}$), the right handed curvilinear frame $\mathbf{u}_i = (\mathbf{e}_u\,, \mathbf{e}_v\,, \mathbf{e}_t\,)$ is defined by

$$\mathbf{e}_t \;= \frac{\mathbf{p}}{|\mathbf{p}|}$$

$$\mathbf{e}_u = \frac{\mathbf{e}_z \times \mathbf{e}_t}{|\mathbf{e}_z \times \mathbf{e}_t|}$$

$$\mathbf{e}_v = \mathbf{e}_t \times \mathbf{e}_u\,, \tag{2}$$

where $\mathbf{e}_z$ denotes the global z axis[4]. Figure 6 shows the curvilinear frame construction for a measurement along a straight line.

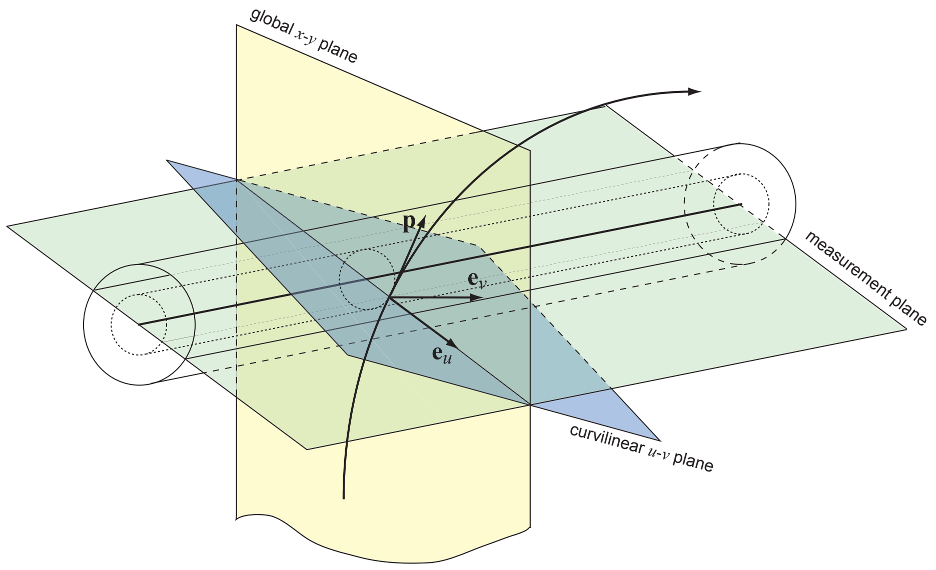


Figure 6: The curvilinear frame $U = (\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_t)$ for a measurement along a straight line. The normalised momentum vector $\mathbf{e}_t = \frac{\mathbf{p}}{|\mathbf{p}|}$ builds the z direction of the curvilinear frame. Its vector product with the nominal global z axis creates the second base of the frame, which lies in the global $x-y$ plane. Finally, the third base vector is chosen in such a way that a right handed coordinate system is well defined.

Let $\mathbf{C}_{li}$ denote the covariance matrix of the initial parameters in the local coordinates of the starting frame, and $\mathbf{J}_{li,ci}$ the Jacobian matrix describing the parameter transformation from the initial local frame li to the initial curvilinear frame ci. The initial covariance matrix expressed in the curvilinear frame $\mathbf{C}_{ci}$ of the starting parameters can then be written as

$$\mathbf{C}_{ci} = \mathbf{J}^{ci,li} \cdot \mathbf{C}_{li} \cdot \mathbf{J}_{li,ci}, \tag{3}$$

with $\mathbf{J}^{ci,li} = \mathbf{J}^T_{li,ci}$ being the transposed Jacobian of the *local-to-curvilinear* transformation.
The transport Jacobian matrix $\mathbf{J}_{ci,cf}$ between the curvilinear frame at start point i and destination point f (i.e. track intersection with the target surface) becomes only dependent on the rotations of the initial and final curvilinear frames with respect to the global frame and the propagation length s. The calculation of this Jacobian such as the coefficients for local to curvilinear frame transformations (and *vice versa*) can be found in the Appendix, Sec. A.5, of this document. The transported covariance matrix at the destination surface after back transformation to the local frame using an appropriate Jacobian $\mathbf{J}_{cf,lf}$ is then given by

$$\mathbf{C}_{lf} = \mathbf{J}^{lf,cf} \cdot (\mathbf{J}^{cf,ci} \cdot \mathbf{C}_{ci} \cdot \mathbf{J}_{ci,cf}) \cdot \mathbf{J}_{cf,lf}, \tag{4}$$

with $\mathbf{J}^{cf,ci} \cdot \mathbf{C}_{ci} \cdot \mathbf{J}_{ci,cf}$ describing the transport along the track and within the curvilinear frame.

Neutral Track Representations The adaption of the `StraightLinePropagator` to neutral parameters was only challenging from a technical point of view, where the same mathematical steps have to be performed on different input classes that share only a high level common base class. This is

[4]In principle, any unique global reference axis can be chosen instead of $\mathbf{e}_z$. The choice of the global z axis in this scope is not arbitrary though, but reflects the detector geometry. It guarantees numerical stability since track directions along $\mathbf{e}_z$ are not expected.

solved by templating the private propagation methods to the different surfaces by the actual charged or neutral parameters type. The `HelixPropagator` simply uses the `StraightLinePropagator` for any parameters transport of neutral parameters or in case of a no-field environment. The transport of the neutral parameters classes is fully supported with ATLAS release 13.1.0.

2.3 Numerical propagation: the `RungeKuttaPropagator` and `STEP_Propagator`

Most of the track parameter propagations to be performed in the ATLAS event reconstruction are within a highly inhomogeneous magnetic field where a global track model can not be used for solving the transport equations. Hence, a fast numerical solution for calculating the intersection of the trajectory with the destination surface is needed. In ATLAS this is realised by two implementations of the `IPropagator` interface, the `RungeKuttaPropagator` and `STEP_Propagator`. Both rely on a fourth order Runge-Kutta-Nystrøm formalism with an integrated adaptive step estimation. The main difference between the two realisations is that the `STEP_Propagator` includes energy loss in the equation of motion and applies corrections to the covariance matrices continuously during the parameter transport along the track. It is designed for the description of a particle that traverses a dense block of material, while the `RungeKuttaPropagator` complies with the classical model of point-like material update on detector layers that is carried out by dedicated `AlgTool` classes in the ATLAS track extrapolation engine. Both `IPropagator` implementations perform the propagation in global coordinates and use common Jacobian matrices for the transformations between the global frame and local surface-attached coordinate systems[5]. The equation of motion of a charged particle with momentum p and mass m through a magnetic field $\mathbf{B}(\mathbf{r})$ can be expressed in many different ways that mostly differ through the parameterisation and choice of the free parameter. For collider experiments a helix-based parameterisation along the arc length s is a good choice since it is not restricting nor favoring any specific particle direction[6]. The equation of motion of a particle with charge q, defined by the Lorentz force, can then — when omitting multiple scattering and energy loss effects — be written as the second order differential equation

$$\frac{d^2\mathbf{r}}{ds^2} = \frac{q}{p}\left[\frac{d\mathbf{r}}{ds} \times \mathbf{B}(\mathbf{r})\right], \tag{5}$$

and, when including an energy loss function $g(p, \mathbf{r})$, as

$$\frac{d^2\mathbf{r}}{ds^2} = \frac{q}{p}\left[\frac{d\mathbf{r}}{ds} \times \mathbf{B}(\mathbf{r})\right] + g(p, \mathbf{r})\frac{d\mathbf{r}}{ds}. \tag{6}$$

Equation (5) and Eq. (6) are the fundamental transport equation used by the `RungeKuttaPropagator` and `STEP_Propagator`, respectively. The calculations are in both cases performed using the Runge-Kutta-Nystrøm method, which is well suited to solve second order differential equations. The basic principle of the Runge-Kutta method can be found in may textbooks [9], an exhaustive review of the used Runge-Kutta-Nystrøm method and the description of error matrix transport (carried out by the Bugge-Myrheim method) for both propagators of the ATLAS track reconstruction is in addition presented in [12].

Both the `RungeKuttaPropagator` and the `STEP_Propagator` stop the numerical iteration when the distance to the surface drops below a certain cut value. For the last step starting at the position $\mathbf{r}_{f-1}$, a simple Taylor expansion to second order is used:

$$\mathbf{r}_{final} = \mathbf{r}_{f-1} + h\frac{d\mathbf{r}}{ds}\,|_{\mathbf{r}_{f-1}} + \frac{1}{2}h^2\frac{d^2\mathbf{r}}{ds^2}|_{\mathbf{r}_{f-1}}, \tag{7}$$

with h denoting the distance to the destination surface at the approach point $f-1$.

Straight and Helical Track Model The `RungeKuttaPropagator` and `STEP_Propagator` are clearly the most flexible propagation techniques in the ATLAS track reconstruction software. It is inert to the Runge-Kutta formalism that in case of a homogenous magnetic field setup, the propagation is carried

[5]The common data classes are located in the shared TrkExUtils package.

[6]In fix target experiments, however, a different choice representing the main particle direction may be taken.

out in one single step, complying a helical model for constant field or a straight line propagation when no magnetic field is present, respectively. This eases the support of neutral track parameters that is completely integrated with ATLAS software release 13.1.0.

3 Navigation

The navigation builds the binding link between propagation and material effects integration in the realisation of the ATLAS track extrapolation engine. During a single extrapolation process, the navigation gathers the information about the traversed material from the reconstruction geometry, the so-called `TrackingGeometry`. The internal connectivity of the `TrackingGeometry` is hereby used to save CPU time expensive global search operations.

Based on the morphology of the ATLAS reconstruction geometry, three different navigation streams can be followed throughout the extrapolation process: navigation within a full static and connective setup, navigation within a dynamic setup and navigation within a detached or unordered setup. The concepts and realisation of the different navigation strategies will be described in the following sections.

Initialisation of the Navigation The first step of the navigation procedure is to determine the starting and destination volume for the track extrapolation. This is done following a three-step procedure that optimises the time spent in the search of the start and end configuration:

- **recall:** in many fitting applications it is necessary to extrapolate from one measurement surface to the successive one along a track. The destination configuration of the previous step is hereby often identical to the start configuration of the next extrapolation process. The `Extrapolator AlgTool` *remembers* therefore the end configuration of the extrapolation process in dedicated private member variables and compares as a first step the starting parameters with the solution of the previous extrapolation call.
- **association:** the ATLAS `TrackingGeometry` provides associations between the objects in the different levels of the geometry hierarchy, i.e. `Surface` objects often point to embedding `Layer` objects that in turn may point to enclosing volumes. This hierarchy model can not be established for all detector structures, but is realised e.g. for a big part of the Inner Detector `TrackingGeometry`.
- **global search:** the global search is the most CPU time intensive navigation step, since it involves the stepping down in the object hierarchy tree of the `TrackingGeometry`. However, it is a backup system for the recall and association attempts and is the only possibility to find the associated start and destination configuration for arbitrary surfaces that are not *a priori* known to the `TrackingGeometry`.

3.1 Navigation between Volumes

The underlying `TrackingGeometry`, characterised at first means by a fully connective set of so called `TrackingVolume` objects, is the key to the navigation process. The `TrackingVolume` class, which is in principle a container of confining boundary surfaces, combines the access to the magnetic field with the information about the traversed material (represented as `Layer` objects or through a dense volume description). The boundary surfaces that build the confinement of the volume extend the common ATLAS `Surface` class and can therefore be coherently used with the propagation `AlgTool`. Since they incorporate information about the attached volumes, the intersection with the boundary surface leads automatically to the next `TrackingVolume` that is crossed by the trajectory. This process is repeated until the destination volume is reached.

The `Navigator AlgTool`, a concrete implementation of the abstract `INavigator` base class, is called by the `Extrapolator` to perform the search for the next `TrackingVolume`. It therefore uses the provided propagator to intersect the appropriate boundary surface of the current `TrackingVolume` and find information about the attached volumes. Each `TrackingVolume` is for this purpose capable of providing an ordered list of boundary surfaces to be intersected, starting from the *best guess* that

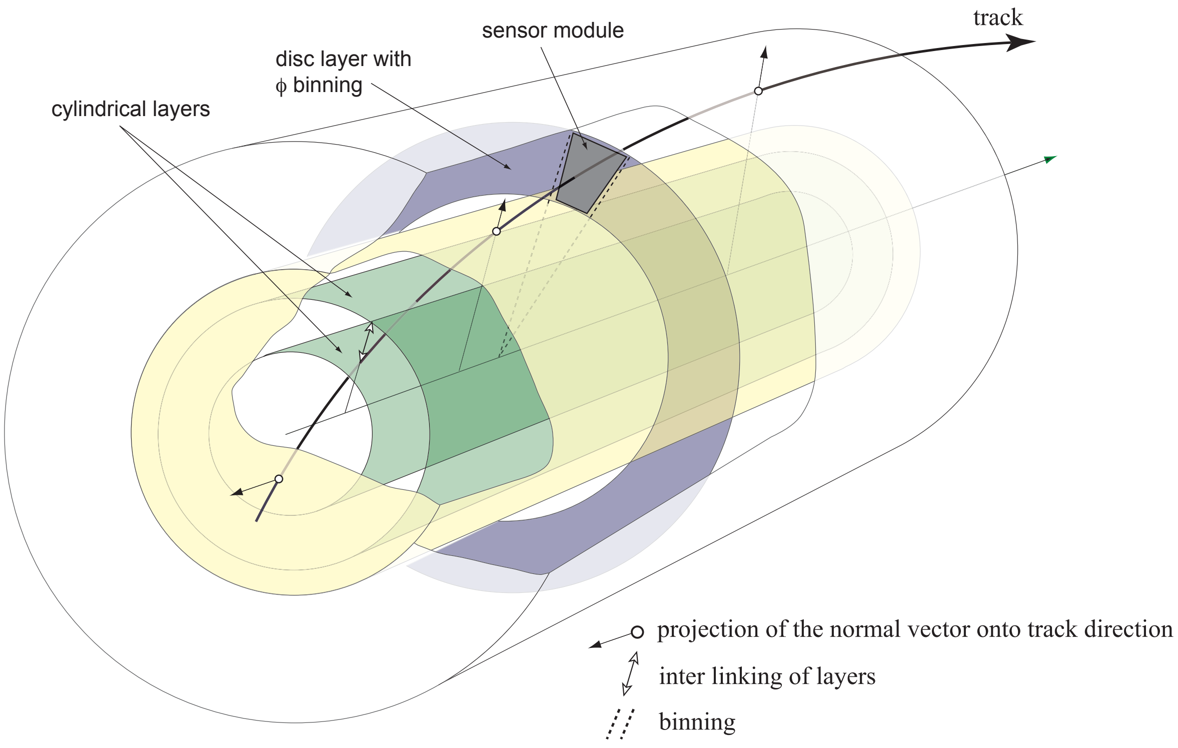


Figure 7: Illustration of the main navigation actions in the static geometry setup: volumes that are via the boundary surface mechanism attached to the currently traversed volume are found by the projection of the track with the according normal vector, layers within a given `TrackingVolume` are inter-linked by pointers and symmetric structures such as sub-surfaces on a `Layer` structure are found by binning methods.

originates from a straight line approximation of the starting parameters in the non bending plane. The provided list of boundary surfaces to be be intersected is processed until the right exit point of the surface is found, for performance reasons only the track parameters and never the associated covariance matrix is transported onto the boundary surface. Performance statistics of the navigation are discussed in further detail in Sec. 6.1.1. Figure 7 summarises the different navigation steps for a full static setup in a sample illustration.

3.2 Navigation between Layers within a `TrackingVolume`

A `TrackingVolume` can contain a static, dynamic or unordered set of `Layer` objects that are suitable for a point-like material update description. Inside a `TrackingVolume`, the navigation has to find the contained layers that are intersected by the trajectory to account for their material budget and apply material effects corrections at the correct point along the trajectory. The intersection with the layer is also needed to be able to use refined local material maps on a single layer.

3.2.1 Static Layer Setup

A static layer setup is realised when the entire description of the material within a `TrackingVolume` is available at the initialisation of the reconstruction geometry and can be ordered uniquely with a given reference direction. The entire default ATLAS ID and calorimeter geometry is described by a static layer setup. `Layer` objects in a static setup are inter-linked by C++ pointers and the navigation from one to the other can be simply done by a `nextLayer()` call, which only requires the track direction in the method signature.

3.2.2 Dynamic Layer Setup

In many configurations the material seen by a track depends very strongly on the track trajectory or a static setup is simply not possible due to the complex geometry structure. In such cases or if the material description is gathered from an outside source (e.g. a material map or material database), dynamically created layers along the trajectory can be taken. A dedicated `IDynamicLayerCreator` interface has been created for this purpose. The `IDynamicLayerCreator` mechanism is also used for global fitting techniques, where the material layers along an entire track have to be known and enter as a whole collection into the global χ^2 function. While the `IDynamicLayerCreator` is defined to provide a material description and still leaves the calculation of the resulting effects on the parameters to the client algorithm, the extrapolation package has been recently expanded with an additional, even more user-open dynamic schema that is designed to provide material effects for the track directly in terms of an energy loss to be applied and a parameter that determines the scattering correction. This second way, realised through the `IMaterialEffectsOnTrackProvider` interface, is mainly targeted at allowing custom integrations of material effects, such as energy measurements provided by the calorimeter.

3.2.3 Unordered layer setup

The `TrackingVolume` also enhances more complex layer structures that can not be easily ordered along reference directions. In this case the navigation has to follow a *trial and error* principle. Hence, the numbers of unordered layers contained by one `TrackingVolume` have to be kept low.

3.2.4 Sub-surfaces on Layers

`Layer` objects can contain ordered sub-sets of surfaces that represent active detector elements. The search for detector elements completes the full predictive navigation provided by the TrkExtrapolation engine and is useful for track predictions and holes-on-track search: the `Surface` class points back to an associated sensitive detector element and since the track trajectory is very precisely known after the track fit a list of geometrically intersected modules can be provided by following the layer and sub-surface navigation (see also Fig. 7). The comparison of the list of surfaces (and thus detector elements) stored on the track with the ones that are geometrically intersected by the given trajectory can be used as a hole definition for tracking. Additionally, the full access to intersected sensitive modules led to the development of the new FATRAS simulation. In both applications, the holes search and the fast track simulation, the local parameters of the trajectory intersection with the layer surface are hereby used for a fast look-up in the binned arrays that hold the sensitive detector surfaces.

3.3 Detached Volumes

The ATLAS reconstruction geometry also describes entities that can not be conveniently integrated into the model of a fully connective `TrackingVolume` set. This is mainly the case for complex structures such as the toroid magnet system in the Muon Spectrometer. A `DetachedTrackingVolume` class which fulfills many requirements of a standard `TrackingVolume`, but is not connected to neighboring volumes via the boundary surface schema, has therefore been created such that they can be contained by a higher level `TrackingVolume`. As a direct consequence, detached volumes require a more CPU time expensive search in the navigation stream. The entry point into a `DetachedTrackingVolume` has to be found first by propagations to the associated boundary surfaces. If a `TrackingVolume` contains more than one detached sub-volume, the sequence of traversing one after the other has to be evaluated first. The `DetachedTrackingVolume` itself contains a complete static setup, such that the fast navigation schema that relies on pre-ordering and boundary surface sharing can be used internally, once a detached object has been entered by the navigation stream.

3.4 Navigation Breaks

The navigation can break in very rare cases mainly due to the reason that provided start and destination information is not compatible, i.e. the trajectory defined by the starting parameters does not intersect the destination surface at all or at least not in the volume that contains the destination surface. The following sources of navigation breaks have been identified and are recorded in case the `AlgTool` is configured to run in a dedicated validation mode:

- **distance increase**: the geometrical distance to the destination surface is checked progressively during the full navigation chain and a navigation break is triggered if this distance increases rapidly at one step, while small fluctuations are allowed. This usually happens if — given a set of starting parameters — it is geometrically impossible to intersect the provided destination surface.
- **oscillation**: the navigation is caught in an oscillation loop, which mostly indicates a geometry setup problem or a curling particle trajectory due to low momentum.
- **loop**: the navigation is caught in a straight loop, i.e. the next assigned volume is identical with the current volume; this indicates usually a geometry setup problem or a wrong propagation solution (i.e. wrong sided cylinder intersection in all recorded cases).
- **no next volume**: the navigation hits the end of the described `TrackingGeometry`; in general, this is also due to incompatible input parameters.

Once a navigation break is triggered, the extrapolation is reset and carried out again without navigation and hence without material effects integration. This behavior can be switched off for validation reasons, such that navigation breaks can be triggered and investigated, see Sec. 6.1.1.

4 Material Effects Integration

A particle that traverses material is subject to both energy loss and directional scattering. While for all stable particles (that are subject of track reconstruction in high energy physics experiments) multiple coulomb scattering is the by far dominant contribution to the directional deflection and thus no distinction between the different particles has to be done, energy loss has to be treated differently for electrons than for all other particles that are in the following also referred to as *heavy particles*. This is, because electrons lose a substantial part of their energy due to *bremsstrahlung*, while for heavy particles ionisation loss remains the dominant effect. As a consequence, a particle definition has to be present in the extrapolation realm to distinguish between the different energy loss mechanism. This is steered by a dedicated `enumeration` type: the so-called `ParticleHypothesis`[7].

Since multiple scattering is a stochastic process with a zero mean deflection it is, in general, not regarded during the transport of the track parameters, but applied as Gaussian or multi-Gaussian process noise addition to the track covariances. Section 4.1.3 will show in the following that the assumption of a Gaussian-distributed multiple scattering noise is valid to a large extent. It will also cover the integration of multiple scattering into the track extrapolation software structure.

Energy loss effects, on the other hand, introduce changes to both the trajectory of the particle and the track uncertainties: the loss of momentum clearly changes the particle trajectory in presence of magnetic field, but since energy loss is a stochastic process it can not be corrected for in a purely deterministic way. In practice, energy loss is taken into account by a deterministic mean (or most probable) value and a relatively small variance to this value. Again, the variance is applied as Gaussian process noise addition to comply with the linear least squares methods predominately used in track fitting. Nevertheless, the Gaussian model is far from optimal for electron energy loss description: Section 4.1.4 will concentrate on both the energy loss treatment for heavy particles and electrons and will discuss the validity range of the given applications.

Since both processes, multiple scattering and energy loss, rely evidently on a correct description of the traversed detector material, providing this material distribution within the reconstruction geometry is an *a priori* requirement, fulfilled by the ATLAS `TrackingGeometry`. The navigation, as described in

[7]The name *hypothesis* is chosen deliberately in this context. In most track fitting applications, the particle identification is not done yet, since it relies on additional information from calorimetry, combined reconstruction or specific interaction signatures with the detector material.

Sec. 3 is responsible for finding and forwarding the appropriate material information to the extrapolation and material integration algorithms. A generic updater `AlgTool`, the `MaterialEffectsUpdator` performs the actual correction of the `TrackParameters` objects.

Point-like and continuous Material Effects Integration The ATLAS extrapolation package enhances two different methods for the integration of material effects during tracking the particle through the reconstruction geometry. A classical point-like material update mechanism is the main material interaction philosophy, carried out through a dedicated `AlgTool`, the `MaterialEffectsUpdator`. It requires a purely `Layer`-based (and thus `Surface`-based) description of the ATLAS detector through a simplified reconstruction geometry. The transport of the track parameters and their associated covariances is hereby fully decoupled from the actual correction of the track representation due to interaction with the detector material. An example for the `Surface`-based modeling of the pixel barrel detector can be seen in Fig. 8.

A second, modern model of continuous material effects integration that embeds the correction as additional terms into the equation of motion can be chosen alternatively, see Sec. 2.3. The continuos material effects integration relies on a `Volume`-based description of the detector through the `TrackingGeometry`. This requires, in general, a more detailed description close to the realistic detector setup or is limited to large structures that can be modeled as one homogeneous block of material[8]. In ATLAS, the continuous integration of material effects is mainly planned to be used for the track extrapolation through the calorimeter and for traversing the large inert material structures in the Muon Spectrometer.

4.1 The `MaterialEffectsUpdator AlgTool`

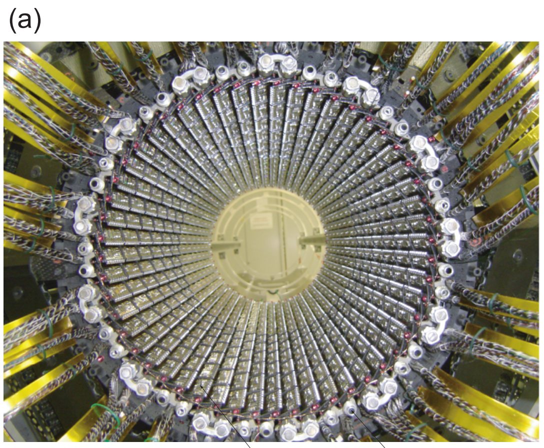
Although both, multiple scattering and energy loss depend on the particle momentum p, they can be treated as two independent (stochastic) processes, since — in general — the energy loss Δ is small compared to the momentum p of the particle. It can therefore be assumed that p is constant during the multiple scattering process, which disentangles the energy loss from the multiple scattering integration. In the realisation of the ATLAS extrapolation package, this relation is respected by two dedicated `AlgTool` implementations that perform the tasks of energy loss respectively multiple scattering calculations separately. The `Extrapolator`, however, only refers to a single `AlgTool`, the `MaterialEffectsUpdator` that uses the other `AlgTool` instances.

4.1.1 Pre-, post- and full update

The material effects integration in the ATLAS extrapolation engine enhances the concept of *pre-*, *post-* and *full* update directives, to account for an optimal spatial position of the material distribution. This is necessary since the reconstruction geometry is a simplified version of the real detector geometry and consists of only several idealised layers that carry a projected material distribution. For the ATLAS silicon detector, for example, the layers of the reconstruction geometry correspond to the actually build barrel cylinders and endcap discs. The largest contribution to the material budget per layer in the silicon detector is the sensor material itself and, furthermore, most of the additional support and cooling structures are located on the back-side of the detector sensor. Deflections and energy loss caused by traversing these modules contribute in a Kalman forward filter consequently only on the successive measurement module — and are applied therefore as *post-update* with respect to the measurement update on the given surface; however, they have to be taken into account before the measurement update in the smoothing or backward filtering process (*pre-update*). The *Transition Radiation Tracker* (TRT) in the ID, on the other hand, can be regarded to have an almost continuous material distribution and is in the reconstruction geometry represented by condensed layers. This description is anyhow only valid in a global picture and the *pre/post* update mechanism is superfluous: a full update is applied when crossing a layer of such a type or any other layer that is not associated to sensitive (tracking) detector elements. Figure 8 shows the necessary simplifications for the pixel

[8]It is of little sense to describe detector parts with a discrete material distribution such as the silicon detector through such a model. The material taken into account would hereby be only valid for traversing the entire volume and is therefore not suitable for track reconstruction, where correct knowledge of the spatial information of the material is essential for a satisfactory description of both multiple scattering and energy loss effects.

barrel reconstruction geometry and the consequences for the material integration in an extrapolation process.



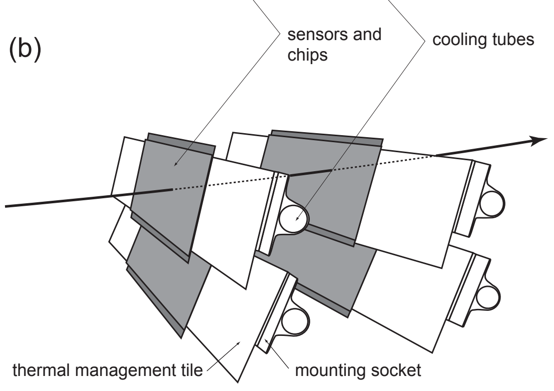
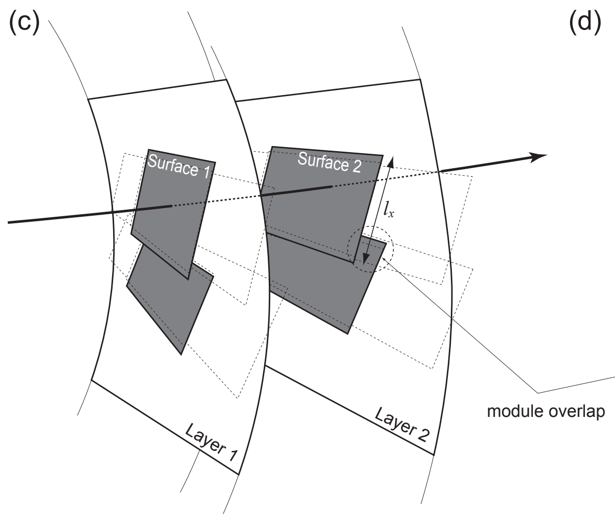
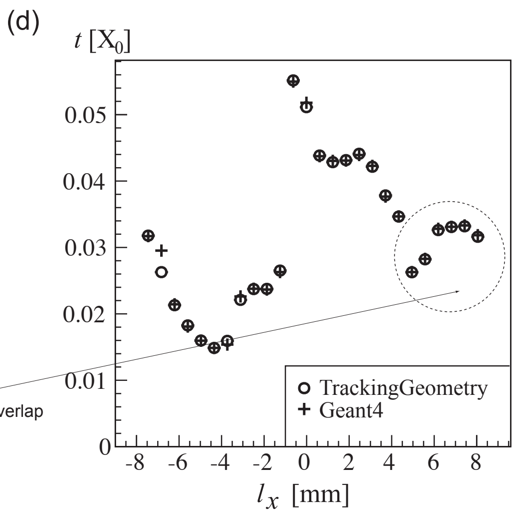


Figure 8: The ATLAS pixel barrel detector as an example for the simplifications done in the reconstruction geometry and their effect on the material integration of the extrapolation package. A photograph of one fully deployed barrel cylinder can be seen in (a), while (b) shows an illustration of the chip and sensor parts which are mounted on staves that include a thermal management tile and a cooling system. In the reconstruction geometry — as illustrated in (c) — the barrels are modeled as cylinder layers and only the detector surfaces are known to full detail. The design of the ATLAS pixel barrel requires that when crossing the layer from the interaction point region, the material of one barrel layer has almost no influence on the measurement of the current, but of the successive layer. In the ATLAS extrapolation package this is taken care of with the *post-*, *pre-* and *full-*update mechanism. Material descriptions exist hereby on both levels, an overall map for the entire cylinder layer and a detailed description of the average material per module. The latter is realised through the *reference material* schema of the ATLAS reconstruction geometry. The reference material map for the pixel module in comparison to the simulation material can be seen in (d).





4.1.2 Correction for the Incident Angle and for Track Bending

The structure of the ATLAS reconstruction geometry and in particular the projective mapping of the simulation material onto the simplified frame of cylinder and disc layers requires two additional corrections to the material distribution that depend on the trajectory properties of the particle and can therefore be only accounted for while performing the actual extrapolation:

- **Correction for the incident angle:** the amount of material seen by a particle when traversing a layer-like structure depends on the relative incident angle of the track with the layer. In the ATLAS reconstruction geometry, particular emphasis has been put on creating a description that is not only valid for particles originating from the nominal interaction point (or center of the coordinate system). That is why — when creating the material maps from the simulation geometry — the material projection of the material seen by a particle coming from the nominal interaction region is recalculated to the fixed frame of cylinders and discs. During the track extrapolation process, the angle of the trajectory with the crossed layer is calculated and the relative correction to the track length in the material applied: **n** and **t** denote in the following the normal vector of the layer and the momentum direction of the trajectory at the intersection point, respectively — both vectors at unit length. The actual crossed path length s through any arbitrarily positioned and oriented layer of thickness d can then be generically expressed as

$$s = \frac{d}{\mathbf{n} \cdot \mathbf{t}}. \tag{8}$$

- **Bending correction:** in a dense volume (e.g. the TRT or calorimeter volumes) that is described through condensed layers in the `TrackingGeometry` a second effect has to be accounted for. Since the mapping of the material distribution is done following a straight line projection, the material budget seen by strongly bent tracks is in general underestimated. This effect can be simply cancelled by introducing a correction factor that is dependent on the transverse particle momentum p_T. ΔR denotes in the following the traversed radial distance in the detector, and B_z the longitudinal component of the magnetic field. The bending radius r of a track can then be written as $r = \frac{p_T}{0.3 \cdot B_z}$. The angle α then describes the opening angle of a transverse track segment and can be expressed as $\alpha = 2 \arcsin(\frac{\Delta R}{2 \cdot r})$. Performing some simple manipulations, the correction factor c_b introduced due to bending can be expressed as[9]

$$c_b = \frac{s}{\Delta R} = \frac{p_T}{0.15 \cdot B_z} \cdot \arcsin \frac{0.15 \cdot B_z \Delta R}{p_T} \approx 1 + \frac{(\Delta R)^2 \cdot B_T^2}{266.7 \cdot p_T^2}, \tag{9}$$

  when the magnetic field is given in Tesla and the momentum is expressed in GeV. Equation (9) shows the dependency of correction the factor to be $1 + 1/p_T^2$, which approaches rapidly towards 1 for transverse momenta bigger than 1 GeV while remaining an about 5% effect at a transverse momentum of 500 MeV, which marks the transverse momentum threshold of many ATLAS reconstruction applications. Figure 9 illustrates the relevant track information that is necessary to calculate the bending correction and shows the correction factor c_b as a function of the transverse momentum p_T.

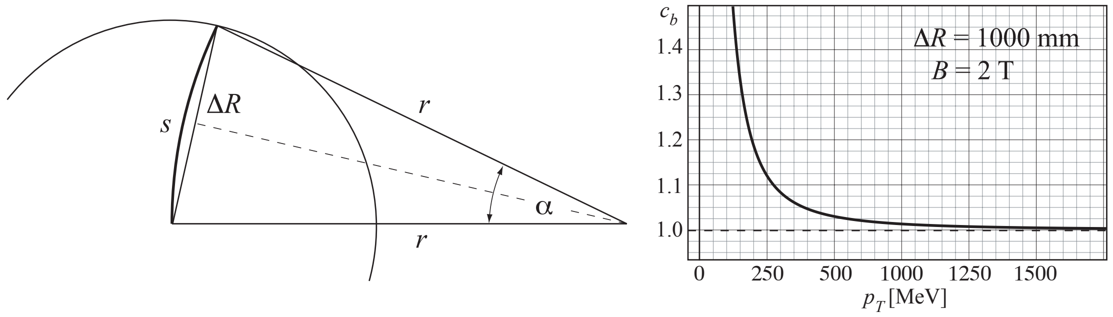


Figure 9: The correction to the material seen by a track crossing condensed layers due to track bending in the magnetic field in an schematic illustration to the left and to the right the dependency of the correction factor on the transverse momentum for a concrete example of traversing a sample distance of 1 m which corresponds roughly to the Inner Detector dimension.

Both the incident angle and the bending correction are matter of the reconstruction geometry, but have to be applied within the material integration during the track extrapolation process. The extrapolation

[9]It is assumed in this formalism that the track enters a volume along a radial direction, which is certainly not true for strongly bent tracks that originate from the nominal interaction point. The resulting correction to ΔR is, however, already taken into account through the incident angle correction when traversing a layer.

engine, however, has been designed to stick to a generic detector model such that no information about the underlying detector specifications can be accessed, while still keeping the maximum performance level. This requirement has been met by defining the corrections as actions known to the `Layer` and `TrackingVolume` that are able to return a `pathCorrection()` or `bendingCorrection()`, respectively, depending on the actual input parameters of the track.

4.1.3 Multiple Scattering: the `MultipleScatteringUpdator AlgTool`

A particle that traverses detector material undergoes successive small angle deflections, caused by multiple (Coulomb) scattering. Given the central limit theorem it can be assumed that the sum of these small variations is Gaussian distributed and symmetrically centered around zero. However, large angle single scattering processes disturb the purely Gaussian probability density function (PDF) and add large non-Gaussian tails. As a rule of thumb, the assumption of the Gaussian character of the underlying PDF is valid to about 98%, being limited to the core region of the distribution.

The integration of multiple scattering effects is handled by a dedicated `AlgTool`, the so-called `Multiple-ScatteringUpdator`. The calculation is done using the well known *Highland formula* [13], which is an empirical adoption of Molière's solution of the transport equation starting from the classical Rutherford cross section of a single scattering process [14].

Highland expanded the original expression given by Molière for the root mean square σ_{ms}^{proj} of the projected scattering angle θ^{proj} with an empirical logarithmic correction term to adopt for the slightly underestimated screening of the nucleus Coulomb potential in materials with lower Z. Furthermore, he transformed — for convenience — the result into a function of the pathlength t in terms of the radiation length X_0 which leads to the well-known expression[10]

$$\sigma_{ms}^{proj} = \frac{13.6\ \mathrm{MeV}}{\beta c p} Z \sqrt{t/\mathrm{X}_0}\,[1 + 0.038 \ln{(t/\mathrm{X}_0)}], \tag{10}$$

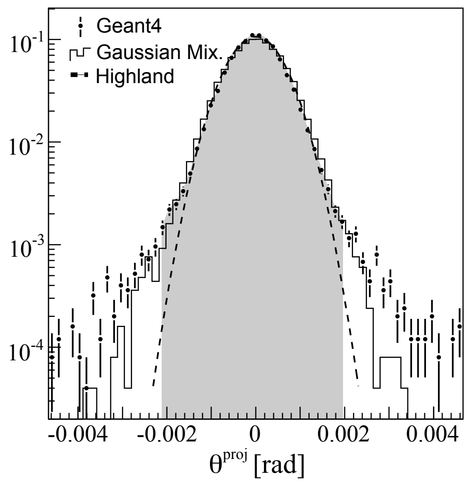
when Z and p describe the charge and momentum of the incident particle, respectively.

Projection Correction and lateral Displacement In the ATLAS track parameterisation the momentum direction is expressed through the globally defined polar angle θ and the azimuthal angle ϕ, see Eq. 1. Since θ already represents a projected angle with respect to the z axis, σ_{ms}^{proj} can be directly applied to the corresponding covariance term, while for the azimuthal angle a correction term of $\frac{1}{\sin\theta}$ has to be applied to the root mean square to account for the out of plane projection.

Another aspect of multiple scattering is that, in general, there exists a correlation between the actual deflection in space θ^{space} and the local coordinates after the scattering process. The local displacement due to scattering is hereby depending on the two projected scattering angles and the thickness of the traversed material. In a `Layer`-based description of the reconstruction geometry, the layer thickness is, however, only a model parameter and has little to do with the actual thickness traversed during the multiple scattering process (in the following referred to as *scattering thickness*). In addition, the `Layer`-based description intrinsically assumes that the material free regions in the according detector volumes are big in comparison to a typical scattering thickness, and the local error on the successive measurement surface is therefore mainly dominated by the directional uncertainties in ϕ and θ. The displacement on the scattering surface caused by the multiple scattering process is therefore omitted in this application. For the continuous integration of material effects, on the other hand, the actual path length s corresponds to the scattering thickness and it is included in the treatment of multiple scattering [12].

Molière's theory of multiple scattering is — when being applied in the small angle assumption — not restricted to a specific particle type nor spin. It is based on the assumption that the deflection of the scattered particle does not change the magnitude of the particle's momentum, or, in other words, it is a pure *elastic* single scattering theory. Rossi and Greisen [15], however, showed that for electrons that traverse a significant amount of material this assumption is not valid anymore since the electron momentum changes substantially due to radiation loss, which is described in more detail in Sec. 4.1.4. This leads to a modification of the momentum dependency from $1/p^2$ to $1/(p_i p_f)$, when

[10]The multiple scattering process itself has little to do with the radiation length X_0 other that both show the same dependency on the atomic number Z and the molecular weight A of the material.



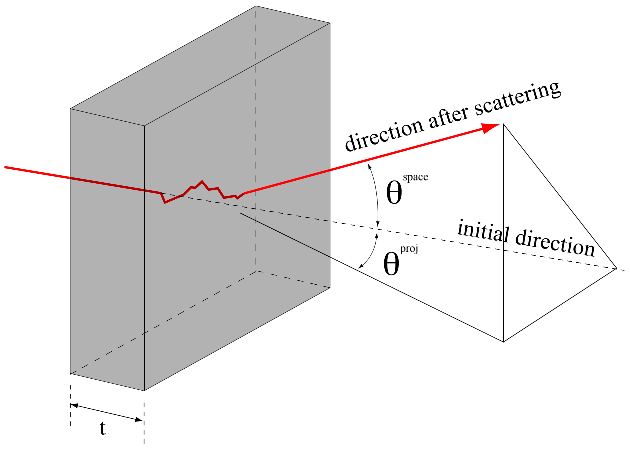


Figure 10: The projected scattering angle distribution θ^{proj} (left) of a muon with 5 GeV that traverses a silicon detector with a thickness t that corresponds to 1% of radiation length X_0 . The distribution (illustrated at standard measure) has been created by Monte Carlo simulation of 25000 muon events using the Geant4 simulation toolkit. It shows the Gaussian distribution originating from the Highland formula (dashed) and the a Monte Carlo Gaussian Mixture model (solid). The shaded area represents a 98% core fraction of the Geant4 distribution. The illustration (right) shows the definitions of the space angle θ^{space} and one projected angle θ^{proj} for an example multiple scattering process.

p_i and p_f denote the initial and final momentum of the multiple scattering process, respectively, and a constant loss assumption is applied[11]. Electron track reconstruction is, on the other hand, mainly determined by the highly non-Gaussian character of the energy loss distribution due to bremsstrahlung that shadows the multiple scattering effects. Identified electron tracks are therefore best fitted with dedicated track fitting techniques. The momentum dependency of the multiple scattering process plays hereby a negligible role.

The integration of such a modified multiple scattering theory for electrons would require the knowledge of the momentum transfer and is not compatible with disentanglement of multiple scattering and energy loss effects. A simplified version of the Rossi-Greisen results, using the $1/p^2$ dependency but sticking to the parameters obtained for electrons can be chosen though for electron multiple scattering in the ATLAS extrapolation package.

Gaussian Mixture Model The Gaussian PDF of stochastic multiple scattering is disturbed by single large angle scattering processes that add significant tails to the distribution of the projected scattering angle. For track reconstruction this effect is negligible since it is small in comparison to loss of accuracy introduced by the necessary simplification of the detector geometry[12]. The ATLAS extrapolation engine that has been initially developed for track reconstruction is also used as the trajectory creation module in the FATRAS simulation and incorporates therefore an additional model for multiple scattering, capable of reproducing parts of the tail distribution. A mixture of two Gaussians with zero mean value, one for the core contribution, one for the tail approximation is used to describe the projected scattering angle distribution

$$f(\theta_{ms}) = (1-\epsilon)\cdot g_0(\theta_{ms};\sigma_{core}) + \epsilon \cdot g_0(\theta_{ms};\sigma_{tail}), \tag{11}$$

where

$$g_0(x;\sigma) = g(x;\mu=0,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{x^2}{2\sigma^2}). \tag{12}$$

[11]More sophisticated models can also be considered where the energy loss function enters directly to the multiple scattering calculation and an integration of the energy loss distribution has to be performed during the scattering process.

[12]It is of second priority to account for a few percent tail effect in the description of the scattering process, when the material distribution is locally not known to such an accuracy.

The model parameters ϵ, σ_{core} and σ_{tail} are hereby depending on the traversed material thickness in terms of the radiation length X_0 and are taken from [16]. Recently [17] even more precise models of the multiple scattering descriptions have been developed that expand the double Gaussian mixture model with a non-Gaussian tail descriptiond. The current ATLAS extrapolation engine has stuck to the Gaussian mixture since it shows satisfactory agreement with the full Geant4 simulation while being lightweight and convenient. A future inclusion of more sophisticated models will however be easily possible by yet another implementation of the `IMaterialEffectsUpdator` interface[13].

Figure 10 shows both the Highland formula application and the Gaussian mixture model as implemented in the `MaterialEffectsUpdator` in comparison with data from a Monte Carlo simulation using the well known and validated Geant4 [18] simulation toolkit. It also illustrates the main definitions used in the calculation of the projected scattering angle.

4.1.4 Energy Loss: the `EnergyLossUpdator AlgTool`

Energy loss of particles traversing detector material occurs due to electromagnetic effects - mainly ionisation (in the order of α^2), bremsstrahlung (order of α^3), direct pair production (order of α^4) and photonuclear interactions; α denotes the fine-structure constant with $\alpha \simeq 1/137$. The PDF $\rho(\Delta)$ (often referred to as *straggling function*) of the energy loss Δ is highly non-Gaussian, but for the use in most track fitting applications an approximation to a Gaussian distribution has to be done.

For heavy particles with masses above 100 MeV peripheric collisions with the detector material — also called *ionisation loss* — dominate the overall energy loss process. The probability for hard collisions with the nuclei of the detector material is suppressed by the factor $\frac{1}{m^2}$ and can therefore be neglected for heavy particles. The electron mass, however, is about 200 times smaller than the mass of the next heavier stable particle and hence interactions with the strong electromagnetic field of the nuclei that cause bremsstrahlung have to be considered. Above a certain energy threshold, bremsstrahlung starts to dominate the energy loss distribution for electrons.

The ATLAS `EnergyLossUpdator AlgTool` performs the energy loss calculation during the track extrapolation process, which depends on the provided `ParticleHypothesis`, the material properties and the kinematic parameters of the particle. The applied corrections are described in the following paragraphs.

Energy Loss of heavy Particles The energy loss Δ of heavy particles in the energy range of final state particles originating from high energy collision experiments is dominated almost entirely by ionisation loss. Although this is a stochastic process that follows a PDF $\rho(\Delta)$, it is justified to treat it as a deterministic mean or averaged energy loss and a relatively small variance. This is, because Δ is usually small in comparison to the particle momentum.

The mean energy loss of a heavy particle per unit length x due to ionisation loss is described by the well known Bethe-Bloch formula [19]

$$\frac{dE}{dx} = \alpha^2 2\pi N_a \lambda_e^2 \frac{Z m_e}{A\beta^2}\left[\ln\frac{2m_e\beta^2\gamma^2 E'_m}{I^2(Z)} - 2\beta^2 + 1/4\frac{E'^2_m}{E^2} - \delta\right], \tag{13}$$

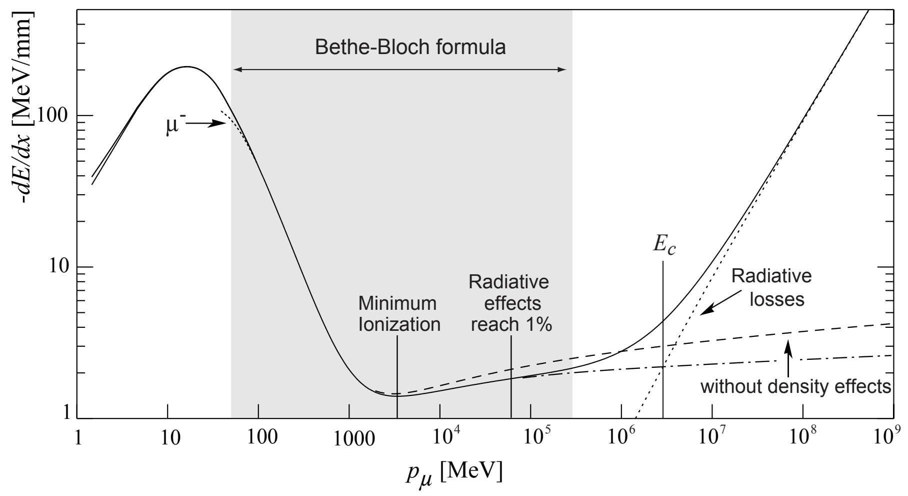
where

N_a	$=$	$6.023 \cdot 10^{23}$, Avogadro's number
Z, A		atomic number and weight of the traversed medium
m, m_e		rest masses of the particle and the electron
β	$=$	p/E, where p is the particle momentum
γ	$=$	E/m
λ_e	$=$	$3.8616 \cdot 10^{-11}$ cm is the Compton wavelength of the electron
$I(Z)$		the mean ionisation potential of the medium,
E'_m		the maximum energy transferable to the electrons of the medium with $E'_m = 2m_e\frac{p^2}{m_e^2 + m^2 + 2m_e\sqrt{p^2+m^2}}$
δ		density correction.

[13]It is worth mentioning that this is one of the biggest benefits of the component software model that has been deployed in the ATLAS track reconstruction.

A parameterisation for the density correction δ and calculated values for various materials can be found in [20].

The Bethe-Bloch formula shows that the mean value of the energy loss depends on the ratio of the particle mass to the particle momentum. The ionisation loss plays therefore often an important role in particle identification, whereas for track reconstruction in the momentum range of interest it is of minor importance. Figure 11 illustrates the defined validity range of the pure ionisation loss assumption given by the Bethe-Bloch formula and shows a comparison of the mean energy loss calculation done within the `EnergyLossUpdator` to values found in literature for various different materials [21].



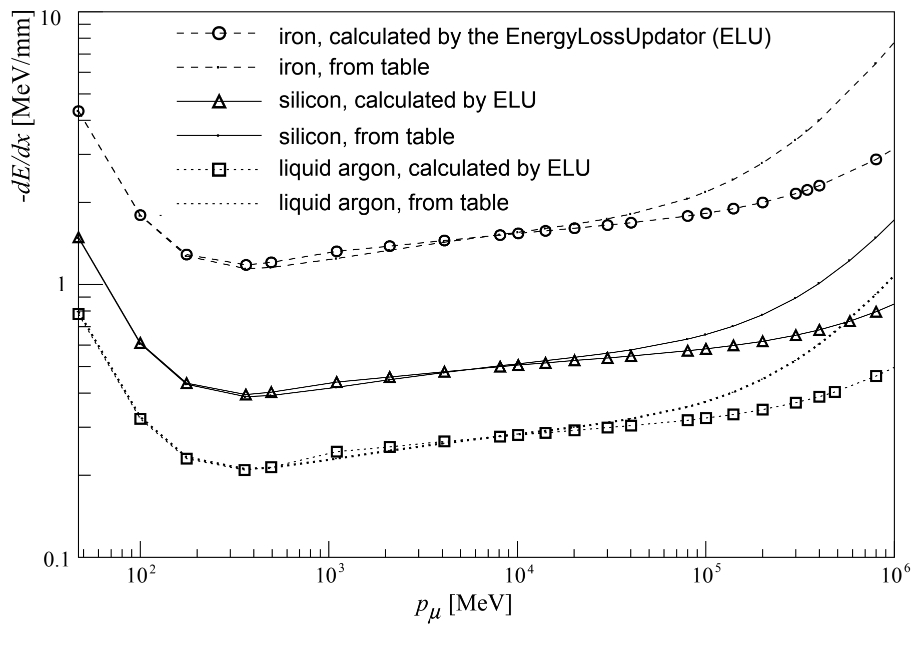


Figure 11: The upper plot shows the mean energy loss per unit length of a muon in copper and is based on [22]. The grey area shows the momentum range where the Bethe-Bloch formula is valid to satisfactory accuracy. The second plot shows a comparison of the mean energy loss of muons in silicon, liquid argon and iron with data taken from [21] and calculated using the `EnergyLossUpdator`. Good agreement can be observed up a particle momentum of about 150 GeV, where radiative effects start to dominate the energy loss distribution. Radiative Effects are not yet implemented in the standard ATLAS `EnergyLossUpdator`.

Landau Distribution and most probable Energy Loss The Bethe-Bloch formula only describes the expected mean energy loss of charged particles due to ionisation. In reality, ionisation loss is a stochastic process that leads to asymmetric fluctuations around a most probable value (MPV). Landau [23] described this distribution[14] and showed that the most probable energy loss ${}_L\Delta_p$ can be expressed as

$$ {}_L\Delta_p = \xi \left[\ln \frac{2mc^2\beta^2\gamma^2}{I} + \ln \frac{\xi}{I} + 0.2 - \beta^2 - \delta \right], \tag{14} $$

with $\xi = ZN_a \frac{k}{\beta^2} t$, when t denotes the thicknesss of the traversed material. It can be shown that in the asymptotic behavior of $\gamma >> 1$ (i.e. $\beta \approx 1$) the density correction δ can be modeled as

$$ \delta \approx 4.447 - \ln \gamma^2, \tag{15} $$

which simplifies Eq. (14) to

$$ {}_L\Delta_p = \xi \left[\ln \frac{2mc^2\gamma^2}{I} + \ln \frac{\xi}{I} - 0.8 + 4.447 \right]. \tag{16} $$

Equation (16) is used in the standard `EnergyLossUpdator AlgTool` for the calculation of the most probable energy loss of heavy particles. Figure 12 illustrates the energy loss distribution for a heavy particle in a silicon layer and includes possible Gaussian approximations that are enhanced by the ATLAS `EnergyLossUpdator`.

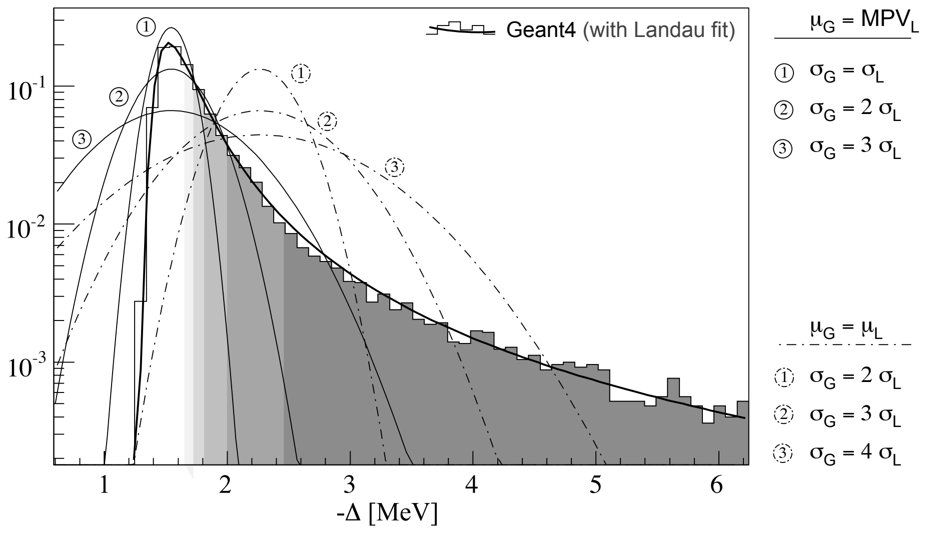


Figure 12: The energy loss distribution at standard measure for single muons with 5 GeV traversing 4.68 mm (i.e. 5% X_0) of Silicon. The distribution has been created with the Geant4 toolkit and fitted with a landau distribution. Different possible Gaussian approximations around the theoretical mean (μ_L, dashed curves) or respectively most probable value (MPV$_L$, solid curves) that can be chosen for modeling the landau distribution are shown. The shaded areas illustrate additional 10% of entries starting from 50% (non-filled area).

Energy Loss of Electrons Electrons lose a substantial part of their energy due to bremsstrahlung, but inonisation loss still remains a contribution to the total effect. The ATLAS extrapolation package is capable of correcting for both, mean ionisation loss and bremsstrahlung even if the ionisation loss can be neglected in many cases. The description of the ionisation loss is slightly modified in the form factors in comparison with the heavy particle case. Energy loss by bremsstrahlung is well described by the theory of Bethe and Heitler [24]. Following a standard notation where z denotes the ratio of the final energy E_f to the initial energy E_i and denoting the amount of material traversed by the particle in terms of radiation length X_0 as t, the PDF of z is given by

$$ \rho(z) = \frac{[-\ln z]^{c-1}}{\Gamma(c)}, \tag{17} $$

[14]Nowadays only known as *Landau distribution*.

where $c = t/\ln 2$ and z is evidently restricted to $z \in (0,1)$.

The average mean (radiative) energy loss per unit length is then given as[15]

$$(dE/dx)_{rad} = -E_i/X_0 \tag{18}$$

From Eq. (18) one can learn that the expectation value for z is $< z >= e^{-t}$ and the variance can be approximated by $\mathbf{var} < z >= e^{-t \ln 3/\ln 2} - e^{-2t}$, which propagates a noise addition of $\sigma^2_{q/p}$ to the covariance matrix of the ATLAS track parameter q/p as

$$\sigma^2_{q/p} = \frac{1}{< z >^2 p^2} \cdot \mathbf{var} < z >, \tag{19}$$

when this kind of update is applied.

The standard track parameterisation used in the ATLAS tracking EDM is defined such that the uncertainties of the track are implicitly assumed to be Gaussian distributed, and it can be shown that the application of the average energy loss described by the Bethe-Heitler formula (including the Gaussian noise addition to the track uncertainties) introduces a strong bias towards too low momentum reconstruction [25]. Figure 13 shows a comparison of the energy loss distributions of a heavy particle (μ) to an electron when traversing the same detector layer.

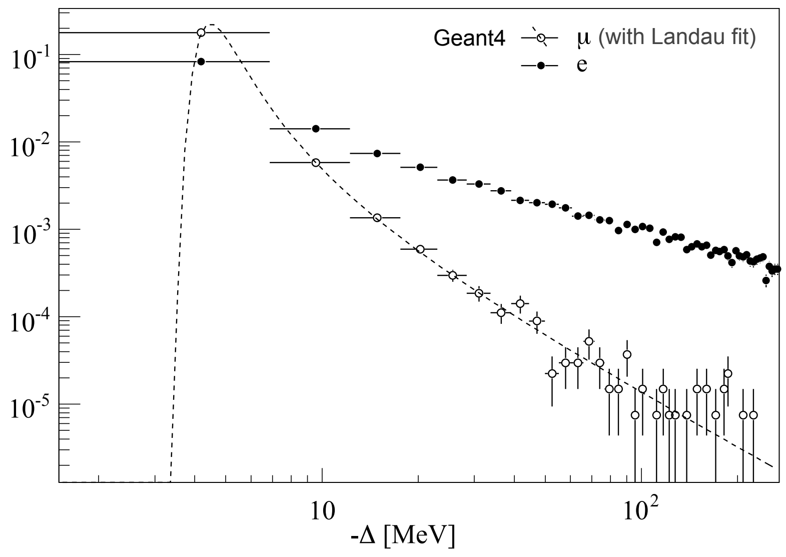


Figure 13: Comparison of muon energy loss to electron energy loss in a silicon layer of 10% X_0 thickness. The particles have been generated using Geant4 and where propagated with a initial momentum of 2 GeV. The muon energy loss distribution follows hereby the Landau distribution, while the electron energy loss distribution is disturbed by the long tail due to radiation loss. This results in a theoretical mean value up to 10 times bigger than for pure ionisation loss.

4.2 Summary of the Material Effects Integration

The ATLAS extrapolation engine enhances different material update mechanisms that have been described within this section. Some of the described options are dedicated to the fast track simulation FATRAS. Table 1 gives — for the convenience of the reader — a summary of the implemented techniques and indicates the configuration flags to be chosen for the various applications. The property flags refer to the `MultipleScatteringUpdator` and `EnergyLossUpdator`, respectively.

The width of the Gaussian approximations to the energy loss functions can be adjusted by specifying one additional property of `EnergyLossUpdator`.

[15]Note that Eq. (18) led to the definition of X_0 through $X_0{}^{-1} \approx 4\alpha r_e^2 Z(Z+1)N \cdot E_i(\ln 183 Z^{-1/3} + 1/18)$, when $N = \rho N_a/A$ is the number of atoms per unit area and r_e the classical radius of the electron.

Table 1: Possible material effects mechanisms that are accessible through the ATLAS extrapolation package, the description is given for multiple scattering (MS) and energy loss effects.

Mechanism	**Equation**	**Particle**	**Property flag**
MS, single Gaussian	Eq. (10)	all	default
MS, double Gaussian	Eq. (11)	all	`GaussianMixtureModel=true`
mean ionisation loss	Eq. (13)	all	default
most probable ionisation loss	Eq. (16)	all	`bool` in method signature
mean radiation loss	Eq. (18)	$e^{+/-}$	`UseBetheBlochForElectrons=false`

5 Extrapolation

Propagation, navigation and material effects integration are steered by one `AlgTool`, the `Extrapolator`. It establishes the simple user front-end and exists in a two-folded way: as a *fully configured* `AlgTool` whose entire setup is done at job configuration via python setup, or as a *strategy pattern* interface where the executing propagation `AlgTool` is part of the method signature.

The following four different extrapolation methods can be performed either in the fully configured or component pattern setup:

- `extrapolate(...)` this is the main extrapolation interface; a starting track representation, a destination surface, a particle hypothesis that determines the type of material interactions, a propagation direction and a boundary directive for the destination surface can be provided. The `extrapolate(...)` interface exists in a two-folded signature: starting from a single track representation, given by the `TrackParameters` class, or from an entire `Track` object. In the latter case, the closest position on the given trajectory is chosen as a starting point of the extrapolation process. This guarantees an optimal accuracy depending on the given track information[16].
- `extrapolateDirectly(...)` this is the direct forward call to the propagation `AlgTool`, it allows to do a simple propagation without navigation and material effects through the `IExtrapolator` interface.
- `extrapolateStepwise(...)` the method signature is identical to the main `extrapolate(...)` method, but the interface method differs by the return type. A step-wise navigation to the destination surface is performed, stepping down to sub-surface level on intersected layers as described in further detail in Sec. 3.2.4. The `extrapolateStepwise(...)` returns the track representations on all crossed active detector surfaces in a `std::vector`. The last element of the vector is the track extrapolation at the destination surface. This method builds the core of the *a posteriori* holes-on-track search.
- `extrapolateBlindly(...)` this method is identical with the step-wise extrapolation, but does not require a provided destination surface. It stops with the last boundary surface of the known `TrackingGeometry` and is used for trajectory creation within FATRAS.

Recently, the extrapolation package has been extended to support neutral track parameterisations and the new track particle base class. This is aimed at extending the current use of the extrapolation package (that is mainly restricted to track reconstruction) and providing the powerful extrapolation methods to hight level event reconstruction algorithms and physics analysis applications.

5.1 Internal Extrapolation Flow

Encapsulated from the user interface, every extrapolation process is performed as a sequence of actions that are realised as `private` method of the `Extrapolator AlgTool`. The first step is the initialisation

[16]In the ATLAS computing model, the track representation loses the hit information when being transformed from the *Event Summary Data* (ESD) the the *Analysis Object Data* (AOD). Consequently, many operations performed on ESD level lead to higher accuracy compared to the similar operation on AOD data.

of the the navigation, performed by the `initializeNavigation()` method: the starting volume and associated layer to the starting parameters are determined as well as the according destination volume. From this point on, the entire extrapolation flow is driven automatically through the volumes that are crossed by the trajectory: the extrapolation is carried out to the boundary of the current volume, until the destination volume is reached. The mechanism of the `BoundarySurface` objects together with the `Navigator AlgTool` are described in Sec. 3, Fig. 14 shows an UML sequence diagram of a sample extrapolation process.

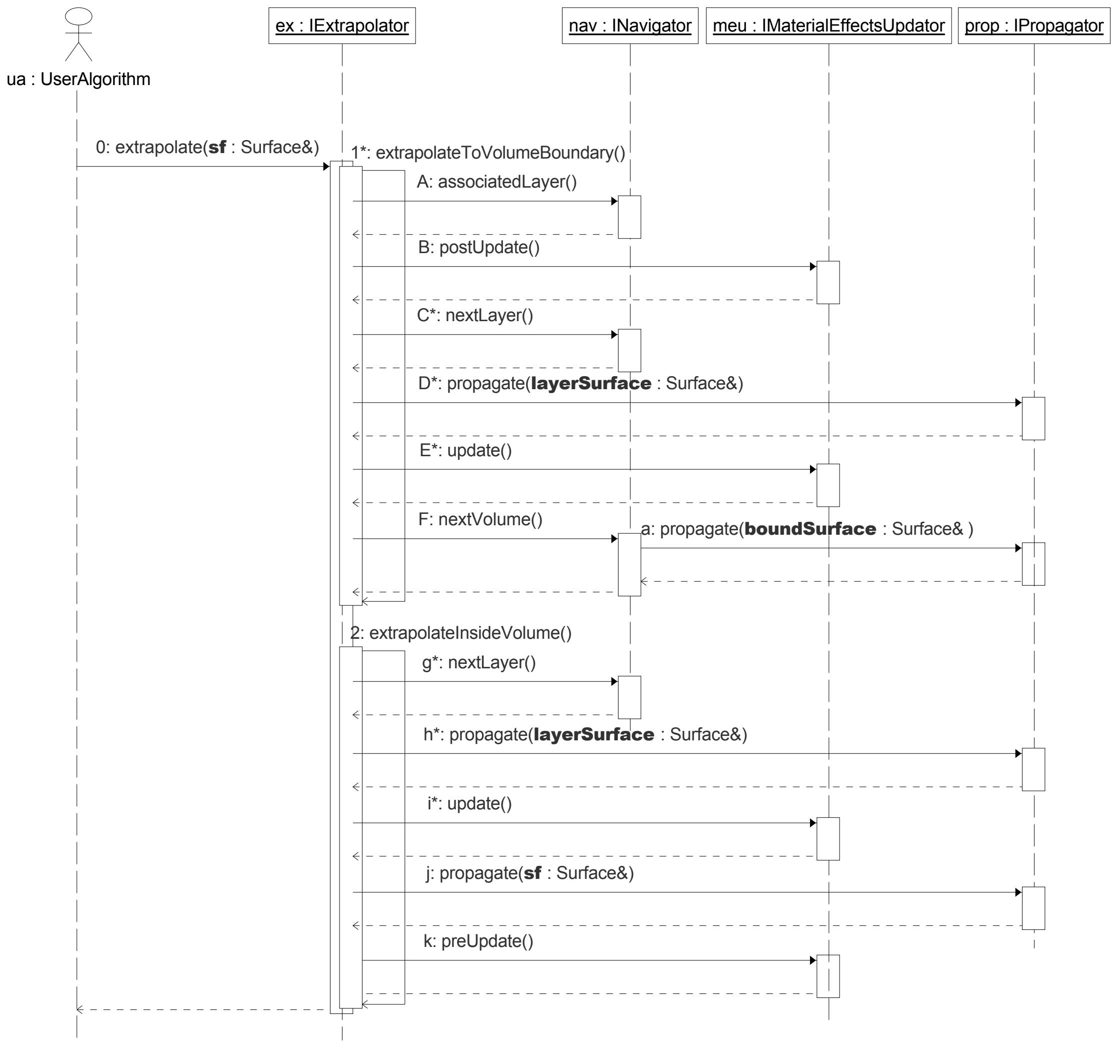

ua : UserAlgorithm
ex : IExtrapolator
nav : INavigator
meu : IMaterialEffectsUpdator
prop : IPropagator
0: extrapolate(**sf** : Surface&)
1*: extrapolateToVolumeBoundary()
A: associatedLayer()
B: postUpdate()
C*: nextLayer()
D*: propagate(**layerSurface** : Surface&)
E*: update()
F: nextVolume()
a: propagate(**boundSurface** : Surface&)
2: extrapolateInsideVolume()
g*: nextLayer()
h*: propagate(**layerSurface** : Surface&)
i*: update()
j: propagate(**sf** : Surface&)
k: preUpdate()

Figure 14: Simplified UML sequence diagram of an extrapolation process; for convenience, the method signatures are mostly omitted and the process is only illustrated for a fully static setup. The only method signatures indicated in the diagram are the different surface types that occur: the destination surface **sf**, various surfaces that represent material layers **layerSurface** and several boundary surfaces labelled as **boundSurface**. Since all surface types extend the same `surface` base class they can be used with the underlying propagation `AlgTool`. The initial `extrapolate()` call to the destination surface **sf** invokes repetitive private `extrapolateToBoundary()` calls until the destination volume is found and the `extrapolateInsideVolume()` succeeds. Inside the volume a sequential extrapolation from one `Layer` class to the next is done.

Volume driven Extrapolation Methods The ATLAS `TrackingGeometry` is put together from volumes of different internal structure; each type triggers a different navigation strategy in the extrapolation process. The most commonly used setup is a fully static description of the detector using layers that are pre-ordered with a given reference direction. The track parameters are hereby propagated from layer to layer and updated according to the material description of the intersected layer. An adjustable and volume dependent maximum number of failed attempts to intersect the successive layer triggers the search for the next volume through intersecting the appropriate boundary surface of the volume. Other parts of the detector may refer to an external material description (e.g. a database, parameterisation) or, when regarding the traverse of the calorimeter, the integration of an actual energy loss measurement could be taken instead of a parametric description through the reconstruction geometry. A dynamic approach has been developed for these cases that allows to create layers for a point-like material integration once the path through a volume is roughly known. It is realised by a dedicated `AlgTool`, the `IDynamicLayerCreator` that can be registered to a volume of the reconstruction geometry and implemented in various different ways. Currently, a wrapper for the material description used in the `Muonboy` reconstruction algorithm [26] that is restricted to the MS exists in a prototype setup.

Additionally, volumes in the reconstruction geometry can be modeled as dense material which allows the usage of the `STEP_Propagator`, see Sec. 2.3. Moreover, detached structures exist in the reconstruction geometry that need specific navigation. All these cases are determined by the structure of the current volume to be crossed.

6 Performance Tests

Track extrapolation processes are very frequently performed by many different modules of the event reconstruction. These client algorithms are immediately affected by the performance and reliability of the track extrapolation engine. Thus, dedicated focus has been drawn on (partly automated) validation and error triggering mechanisms during the design phase of the TrkExtrapolation realm. One major part of the overall track extrapolation quality is the correct description of the underlying detector material by the reconstruction geometry, which is not in particular subject of the extrapolation process itself and will not be covered in this section. A detailed consideration and reflection of this topic can be found in [7].

Both the reliability and the accuracy (or quality) of the track parameter extrapolation have to be investigated. Whereas the first part is somewhat trivial since it simply includes consistency checks and the monitoring of a high number of extrapolation processes, the second task appears to be less trivial as it first occurs. The quality of the extrapolation solution is determined by the correct handling of material interactions and the precision of the mathematical transport of both parameters and covariances through the magnetic field. Latter can be done by comparing the calculated transport Jacobian matrices with numerically evaluated ones, which will be shown in Sec. 6.2. The numerical estimation of the transport Jacobian relies on the correct propagation of the track representation, which can not easily be validated, since the true solution of the intersection is not known. One is thus restricted to compare different modules and propagation techniques amongst each other and to perform self-consistency checks with forward-backward extrapolations.

6.1 Stand-alone Extrapolation Tests

The resulting track parameters resolution is evidently the ultimate validation of the the track extrapolation. However, it accumulates all different aspects of track reconstruction and thus complicates the monitoring of single effects. It is sometimes useful to factorise the various parts of the track reconstruction into smaller modules that can be investigated in a controlled environment. The ATLAS extrapolation package offers dedicated algorithms to monitor special aspects of the extrapolation process. Most of the `Algorithm` classes designed to perform sample extrapolations are located in the TrkExAlgs package of the Tracking repository.

6.1.1 Self-consistency and Navigation Performance

A stable and reliable navigation process through the `TrackingGeometry` is necessary to assure the correct material integration during the transport of the track parameters. While in the track reconstruction the `Extrapolator` is configured to switch to a direct extrapolation without navigation in case that a navigation break has been triggered (see Sec. 3.4), a dedicated `Algorithm` exists for validation purpose that stops the navigation process in such a case and records the specifications of the input parameters and the break condition. This algorithm also allows the monitoring of the boundary surface mechanism, since it records the necessary switches to the second or third boundary hypothesis when exiting a `TrackingVolume` of the reconstruction geometry. It can be shown that the navigation is reliable at highest level, starting from compatible input parameters (i.e. after testing that a geometrical intersection of the trajectory with the given destination surface exists) only one in about 40000 test extrapolations fails due to a navigation break[17]. The extrapolation tests have been carried out incorporating full material effects integration in a momentum range between 500 MeV and 150 GeV . For each randomly generated destination surface within the Inner Detector volume, the transport of the track parameters has been performed before the parameters on the destination surface have been back extrapolated to the starting position. Energy loss is applied in the forward direction, while the deposed energy is again added to the $\frac{q}{p}$ parameter during backward extrapolation. By this, the self-consistency of the material effects integration can be checked accordingly. Figure 15 shows absolute or relative errors for track parameters after the extrapolation to the destination surface and back extrapolation to the initial starting frame. One other aspect can be monitored during theses tests: if the `Navigator AlgTool` is configured to run in validation mode, it records the performance of the entry, respectively exit surface estimation of the various volumes. It can hereby be shown that within typical extrapolation processes, the straight line approximation that is used to estimate the hit volume surface is to more than 99.8 % correct.

6.2 Validation of the Covariance Propagation

The correct transport of the track parameters errors is essential for a good tracking resolution, in particular when the extrapolation is used with progressive fitting techniques that rely on the transported covariances[18]. The validation of the covariance transport through the propagator implementations is therefore inevitable. A dedicated validation algorithm has been established that is capable of comparing the transport Jacobian matrix as calculated by the `IPropagator` implementations with numerically evaluated ones. The numerical derivatives are hereby calculated by using the method of Ridders [27] that is based on a symmetric derivative

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \tag{20}$$

and aims to calculate $f'(x)$ in the limit $h \to 0$. Evidently, Eq. (20) can not be evaluated for this limit, some approaching procedure to the limit value has therefore to be taken. The basic idea of Ridders method is to decrease the parameter h successively and to evaluate the derivative for each case. The limit value is then calculated by a polynomial extrapolation towards $h = 0$. Figure 16 shows the relative residuals of the Jacobian matrix entries

$$r_{jk} = \left(\frac{\mathbf{J}^{calc}_{if} - \mathbf{J}^{num}_{if}}{\mathbf{J}^{num}_{if}} \right)_{jk} \tag{21}$$

between the numerically evaluated and the provided Jacobian matrices, but is probably not very intuitive for representing the quality of the covariance transport, since it does not reflect the different relevance of the various coefficients. Naturally, some components of the covariance matrix play a more important role than others in the track fit, since some track parameters pairs are stronger correlated then others (this is mainly determined by the detector and magnetic field setup). The pure comparison of all Jacobian coefficients, however, does not distinguish between *relevant* and *non-relevant* components. A second, more striking way of showing the quality of the covariance transport

[17]In a typical reconstruction job this number is evidently larger, since the compatibility test can not be *a priori* done. During the track reconstruction the assignment of a hit to a given trajectory is, in general, based on a cruder assumption that is given through the pattern recognition.

[18]In least squares fitters, a crude prediction of the covariance matrix is, in general, sufficient, since it is mainly used to evaluate the compatibility of the hit with the given track hypothesis but does not enter the fit *per se*. However, the transport Jacobian matrix that relates the fitted parameters with the track parameters on the various measurement surfaces is indeed needed for the least squares fit.

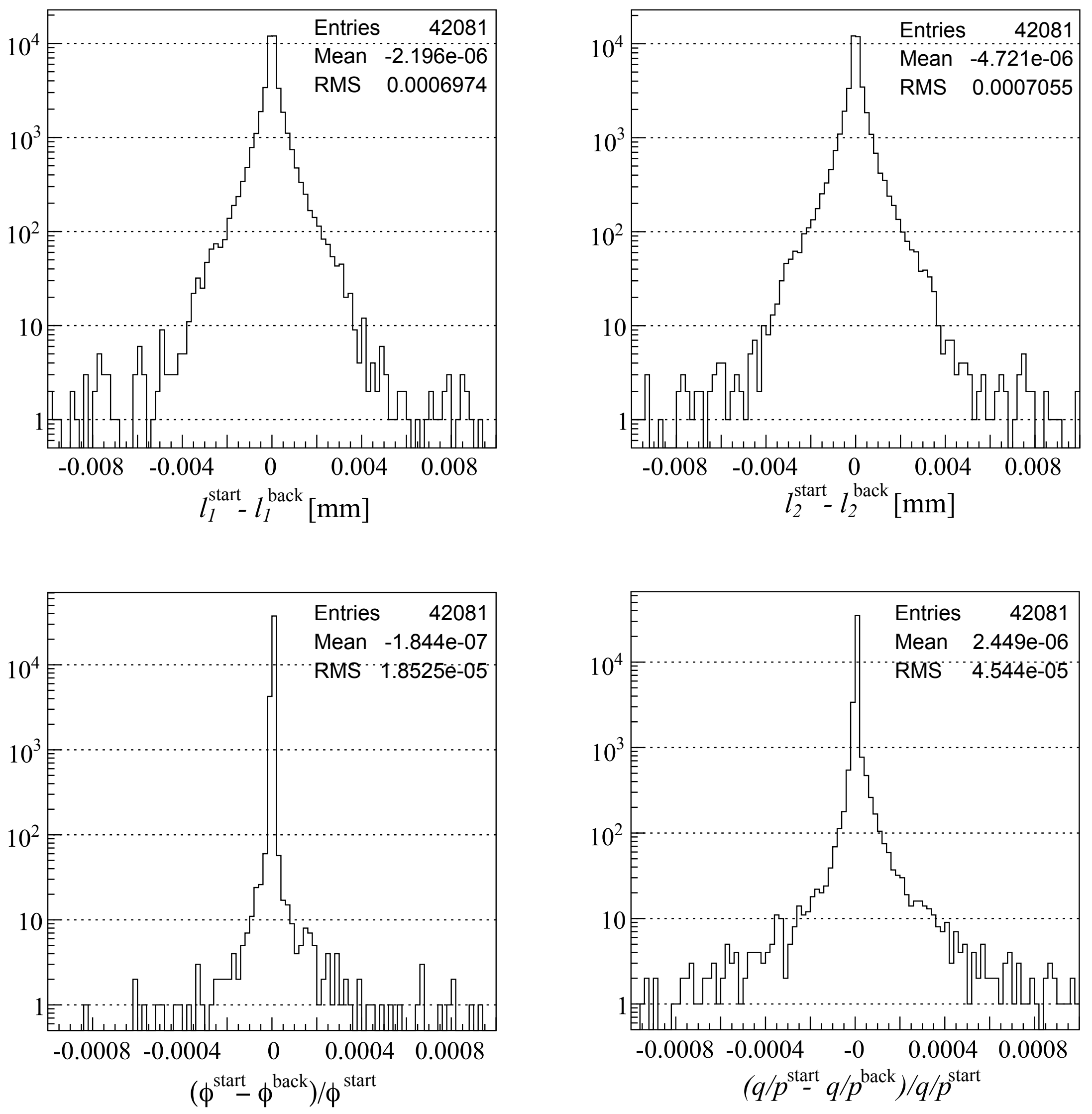


Figure 15: The absolute respectively relative error of about for sample extrapolations within the Inner Detector including full navigation and material effects integration. About 40000 extrapolations for the interaction region (smeared start parameters) to randomly distributed planar surfaces in the Inner Detector and back to the start surface are performed. The two upper plots show the absolute difference of the local coordinates of the start parameters with the back propagated ones. The latter plots show the relative error on the initial ϕ respectively $\frac{q}{p}$ values, the start parameters have been generated equally distributed over the entire azimuthal range and in the momentum interval between 500 MeV and 150 GeV.

is to check the parameter pull distributions

$$f(x_i) = \frac{(x^{rec} - x^{true})_i}{\sigma_i} \tag{22}$$

for the five parameters after the track fit. Unfortunately — when using full detector simulation — the pull distributions are mainly dominated by the quality of the material description used in the track fit and the error estimation based on clusterisation. The pure parameter transport is thus, in general, shadowed by these more significant effects. The fast track simulation FATRAS, however, allows to create tracks with fully controlled input in both material description and hit errors. Figure 17 shows the pull distributions of the track parameters after refitting of 50000 single tracks simulated without material effects and purely gaussian cluster errors in the ID barrel ($|\eta| < 2.5$) using FATRAS .

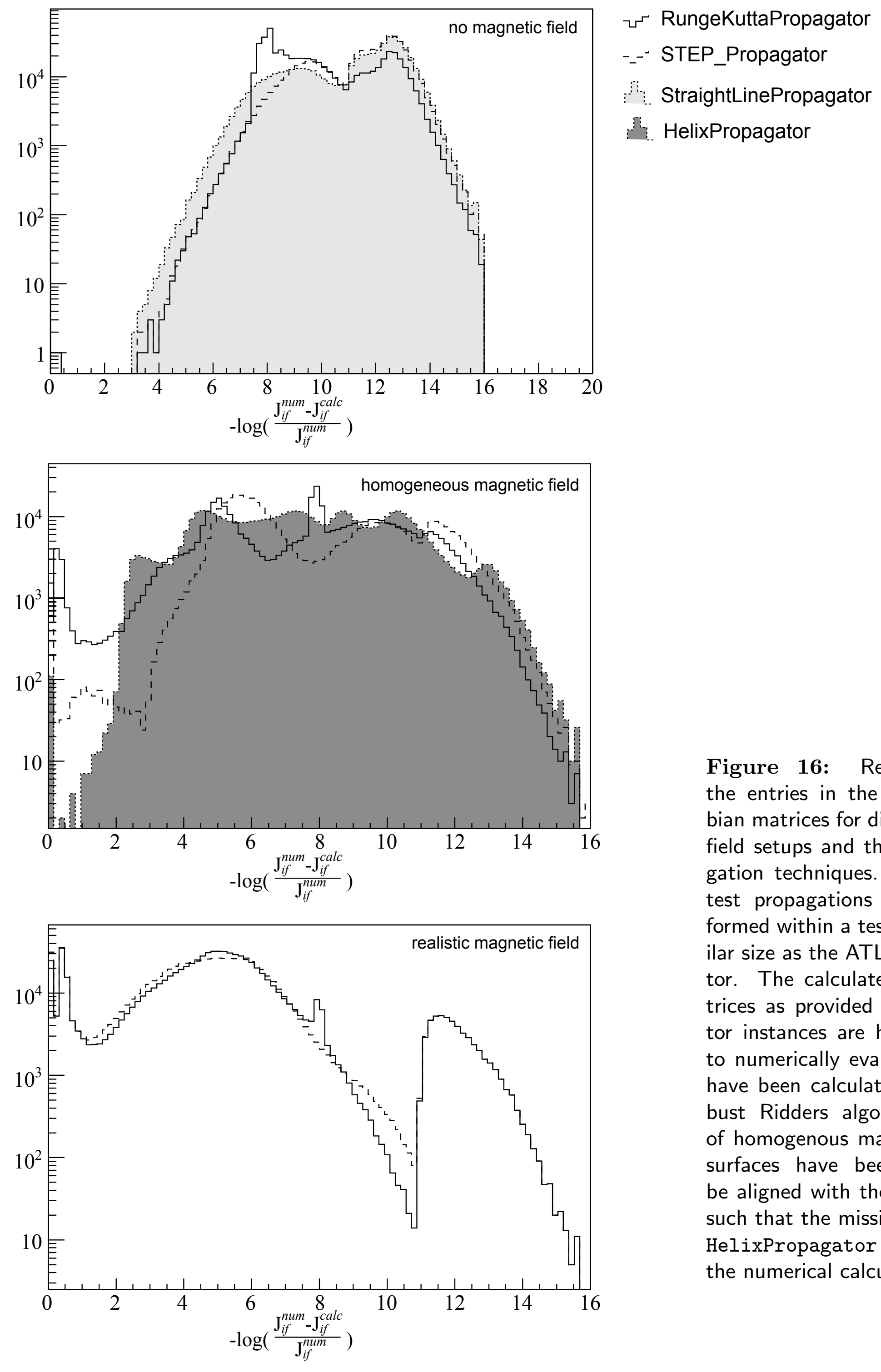


Figure 16: Relative errors of the entries in the transport Jacobian matrices for different magnetic field setups and the various propagation techniques. Fifty thousand test propagations have been performed within a test volume of similar size as the ATLAS Inner Detector. The calculated Jacobian matrices as provided by the propagator instances are hereby compared to numerically evaluated ones that have been calculated using the robust Ridders algorithm. In case of homogenous magnetic field, the surfaces have been restricted to be aligned with the magnetic field, such that the missing tuning of the `HelixPropagator` does not affect the numerical calculation.

6.3 Timing Performance

Since the extrapolation of track parameters is a very frequent process in the event reconstruction, it is of particular interest to be keep the CPU time contribution at minimal cost. The highest relative frequency of extrapolation calls in the event reconstruction is within the track fit, where every hit causes — depending on the fitting technique — usually at least two extrapolation steps. The optimisation of the timing performance of the track extrapolation package can therefore be best evaluated in the minimisation of the contribution to the track fitting applications.

In the standard ID event reconstruction of 10 $t\bar{t}$ events using the *New Tracking* (NEWT) [29] reconstruction chain, the `Extrapolator` is called about 80 000 times, predominately in the track fit;

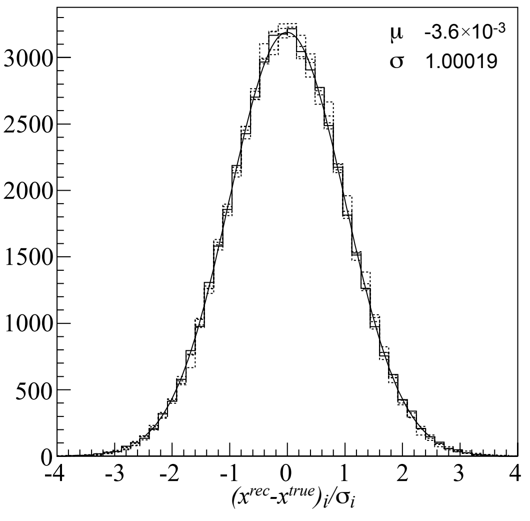


Figure 17: Perigee parameter pull distributions of 50000 tracks that have been simulated and refitted without material interactions and assuming Gaussian distributed cluster errors. All five track parameters are shown, as an example the pull distribution of transverse impact parameter d_0 is fitted with a Gaussian curve. The `RungeKuttaPropagator` has been used for these test.

the `Propagator AlgTool` is called additionally ten times more often in the pattern recognition. Table 2 presents an overview of the number of method calls and relative timing contributions of the extrapolation package per event.

The contribution of the extrapolation process to the track fit is obviously dependent on the used implementation of the ATLAS `ITrackFitter` interface. For the standard ATLAS Kalman filter, the extrapolation accounts to about two thirds of the overall time spent in track fitting during the entire event reconstruction. This number, however, includes the large number of track fits in the ambiguity solving process where many track candidates describe fake tracks and hence the navigation between the given measurement surfaces is not always straight-forward. It can be shown that for the refit of already found and resolved tracks the contribution of the extrapolation drops below 40 %. When using the global χ^2 implementation of the standard ATLAS track fitter interface, the relative contribution to the track fit is in general higher, since for each measurement the extrapolation is at least done three times to calculate the track derivatives numerically stable on the measurement surface. Recently, the propagator interface has been expanded to provide the used transport Jacobian to the client algorithms and thus the number of performed extrapolations in the least squares fit could be reduced to a single one per track. Additionally, this led to an increased stability of the track fit.

Table 2: Relative timing contribution to the event reconstruction measured for 10 $t\bar{t}$ events in the Inner Detector New Tracking reconstruction for the extrapolation components in the pattern recognition and track fitting process. The number marked with (★) are contained in the caller contribution.

Method	**Calls**	**Caller**	**Time/Evt [%]**
`RungeKuttaPropagator::propagate()`	789 238	pattern recognition	8.39
`Extrapolator::extrapolate()`	53 714	track/vertex fitting	2.77
`Extrapolator::extrapolateStepwise()`	26 549	*holes-on-track* search	1.56
`RungeKuttaPropagator::propagate()`	118 189	`Extrapolator`	1.63★

Internal timing statistics Two main contributions to the total time spent during a extrapolation process exist: the transport of the track parameters (and in particular their errors), such as the navigation that is responsible for finding the next volume to traverse. The modification of track parameters due to material effects is a negligible factor in the overall timing of the extrapolation engine. In the current realisation of the ATLAS `Extrapolator AlgTool` both steps are performed with the underlying propagation engine. The relative time contribution of the navigation (including the transport of the navigation parameters to the boundary surface) to the time spent while traversing a volume is about 25 %, but since the fast straight line test performed by the boundary surface mechanism fails only in less than 0.1 % this number can be clearly reduced in a future evolution of the extrapolation engine. Figure 18 shows a caller-diagram of the main extrapolation method in a static volume setup.

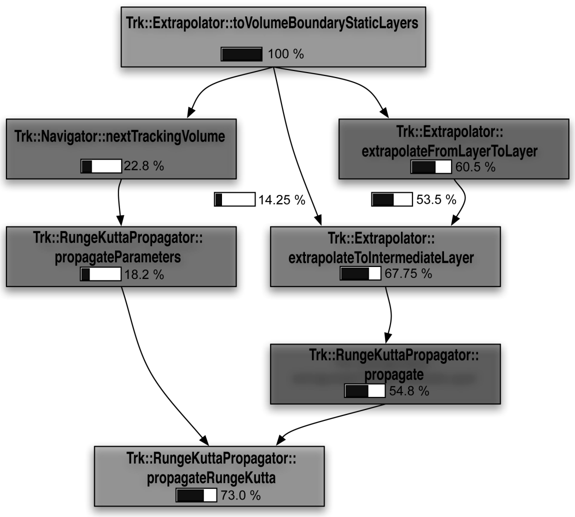

Trk::Extrapolator::toVolumeBoundaryStaticLayers
100 %
Trk::Navigator::nextTrackingVolume
22.8 %
Trk::Extrapolator::
extrapolateFromLayerToLayer
60.5 %
14.25 %
53.5 %
Trk::RungeKuttaPropagator::
propagateParameters
18.2 %
Trk::Extrapolator::
extrapolateToIntermediateLayer
67.75 %
Trk::RungeKuttaPropagator::
propagate
54.8 %
Trk::RungeKuttaPropagator::
propagateRungeKutta
73.0 %

Figure 18: Simplified caller diagram for 500 sample extrapolations within the Inner Detector volume. The graph shows the contribution of navigation and actual extrapolation to the total time spend in the extrapolation to a volume boundary. In the current realisation of the ATLAS extrapolation package, the navigation is also performed using the underlying propagation engine. Encouraging results of stand alone navigation tests — see Sec. 6.1.1 — indicate however that a satisfactory performance can be also achieved by the simple straight line assumption of the boundary surface mechanism. It can be seen that when using the default `RungeKuttaPropagator` about 75% of the extrapolation time is spent in the propagation of track parameters and their errors, while the main contribution to the Runge-Kutta integration is the multiple lookup of the magnetic field value.

7 Conclusion and Outlook

During the last three years a powerful and well performing new extrapolation package has been developed in full coherence to the new ATLAS reconstruction geometry and integrated in the new modular software structure of the ATLAS track reconstruction. It serves as a central part of many applications in the event reconstruction including pattern recognition, track and vertex fitting and combined reconstruction. Several different propagation techniques have been integrated and the modular design facilitates future adaption and extension.

The new extrapolation engine extends the pure functionality needed for track and vertex fitting with a predictive navigation that can be used — together with the underlying tracking geometry — for holes-on-track search or for the fast track simulation FATRAS.

Large scale event reconstruction using the new extrapolation package has been carried out using Monto Carlo simulated data and taken data from the CTB 2004 and the commissioning runs using cosmic rays [28].

7.1 Outlook

An improvement in terms of a lower CPU time consumption of the extrapolation process can be achieved by simplifying the navigation process to directly use a straight line approximation for most of the cases, while including a fallback solution to full propagation if the straight line case fails to determine the correct exit surface, as shown in Sec. 6.3. An alternative approach could be to evaluate the usage of the recently integrated less complex `IIntersector` implementations, that exist as prototype versions in the ATLAS software release 13.1.0.

Recently, the propagation interface has been expanded to provide the transport Jacobian matrices to the client algorithms. This is in particular important for global track fitting techniques, but has not been fully propagated through the extrapolation interface. A future adaption foresees the access to the full transport information (including the noise terms due to material effects integration) through a modified `IExtrapolator` interface.

The ATLAS extrapolation engine has become a widely used tool for many different applications and is currently being used with various different geometry setups and navigation philosophies. This resulted in a rather extensive interface that concentrates these various different tasks. Current work is ongoing to split these different tasks that can be associated to with the different geometry setups of static, dynamic and detached volumes structures in the extrapolation process into separate `AlgTool` classes that comply with the same interface definition.

A Appendix

A.1 Conventions and Typesetting

The following type setting conventions are followed throughout this document: Software packages within the ATLAS offline software repository [30] are written in Sans-sarif face, C++ or python class names are written in `Courier` face. Namespace definitions as used in the software repository are omitted in this document for readability. An exhaustive list of software packages and their location within the ATLAS software repository can be found in Tab. 3.

Table 3: Software leaf packages to be found in the Tracking/TrkExtrapolation CVS repository that are described or referred to within this document.

Leaf package	**Brief description**
TrkExAlgs	test and validation algorithms
TrkExExample	test steering package
TrkExSlPropagator	propagation `AlgTool` with a straight track model
TrkExHelixPropagator	propagation `AlgTool` with a helical track model
TrkExSTEP_Propagator	propagation `AlgTool` with material interactions
TrkExRungeKuttaPropagator	propagation `AlgTool` for inhomogeneous magnetic field
TrkExInterfaces	concentration of interfaces
TrkExDynamicLayerCreator	`AlgTool` to provide layer information to global fitters
TrkExTools	extrapolation `AlgTool`
TrkExUtils	shared utility classes

Typesetting of mathematical Formulas and geometrical Conventions Three different types of three-dimensional frames are used within this document or are referred to by the TrkDetDescr container:

- a coordinate system corresponding to the description of the magnetic field and the detector geometry, referred to as *global frame.*
- a cartesian frame moving along the track, the so-called *curvilinear frame*; the tangential momentum vector of the track builds the z-axis of this frame, x-axis and y-axis are then constructed with an additional global constraint.
- three-dimensional cartesian frames, different from the global frame, mainly attached to surfaces and volumes.

The standard cartesian set $\mathbf{e}_i$ of unit vectors describing the tracking frame are indicated as

$$E = (\mathbf{e}_x\,, \mathbf{e}_y\,, \mathbf{e}_z\,). \tag{23}$$

The standard base of the cartesian curvilinear frame is, for convenience, indicated as

$$U = (\mathbf{e}_u\,, \mathbf{e}_v\,, \mathbf{e}_t\,). \tag{24}$$

Finally, the standard base $\mathbf{h}_i$ of a three-dimensional cartesian frame different different from the tracking frame is noticed as

$$H = (\mathbf{h}_x\,, \mathbf{h}_y\,, \mathbf{h}_z\,). \tag{25}$$

In conjunction with a `Surface` object, this frame is sometimes referred to as *helper frame*, since it builds the carrier of the two-dimensional intrinsic surface frame (also called *local* frame).

A three-dimensional vector in either of the two frames is indicated by bold. Its expression with respect to the different frames is done using the standard base sets:

$$\mathbf{a} = a_x^e \mathbf{e}_x + a_y^e \mathbf{e}_y + a_z^e \mathbf{e}_z = a_u^e \mathbf{e}_u + a_v^e \mathbf{e}_v + a_t^e \mathbf{e}_t = a_x^h \mathbf{h}_x + a_y^h \mathbf{h}_y + a_z^h \mathbf{h}_z \, . \tag{26}$$

A.2 Extrapolation to Distorted Surfaces

In reality, many surfaces in the ATLAS detector that are modeled in event simulation and reconstruction as ideal geometrical objects will be deformed or distorted with respect to their mathematical description. Detector straws or wires in drift tube detectors (such as the TRT in the ID or the *Monitored Drift Tubes* (MDT) in the MS) will be deformed due to gravitational wire sagging; planar surface may be bent due to gravitation for extensive objects or simply due to mounting tension. Some of these effects will be dealt with at the stage of hit calibration when a parametric or projective approach is of satisfactory accuracy, but in particular for large objects, such as the long MDT tube wires, the geometrical distortion has to be taken into account during the extrapolation process. The ATLAS `TrackingGeometry` provides hereby a dedicated `DistortedSurface` base class; concrete surface types extend this base class and the appropriate surface class from the TrkSurfaces repository. The extrapolation process to a distorted surface is then realised by a two step propagation to the final destination surface. Since the `DistortedSurface` types extend the common surface class it can be naturally used with the propagator to find an intersection with the nominal or non-distorted surface. In the following, a corrected surface in position and rotation of same type is created dynamically depending on the intersection solution of the first propagation. The final propagation is then performed to the corrected and more realistic surface position. Figure 19 shows an illustration of an example propagation to a wire surface with gravitational sagging correction (`SaggedLineSurface`).

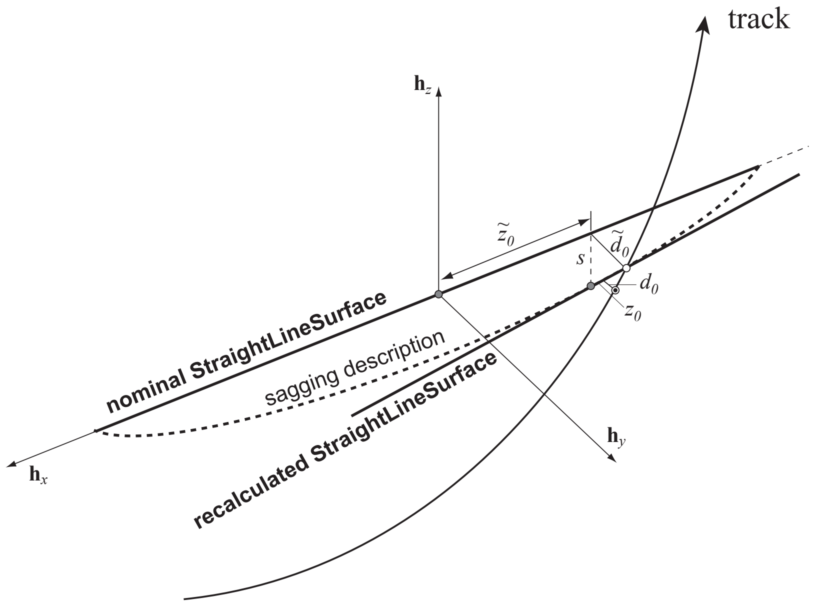


Figure 19: Illustration of a propagation to a wire surface with gravitational sagging correction: a first propagation to the nominal `StraightLineSurface` is performed which yields to the local coordinates $\tilde{d}_0$ and $\tilde{z}_0$. Depending on the longitudinal position on the line and the line orientation, the corrected `StraightLineSurface` is created and the final propagation performed.

A.3 Intersection formulas for straight line propagation

The following sections summarise the formulas used for the calculation of trajectory intersections with destination surfaces of different types in the `StraightLinePropagator`.

A.3.1 Line-Line Approach

The problem of finding the closest approach between two lines in three-dimensional space — one line describing the particle trajectory, the other one a straw measurement or the z axis for a perigee representation, respectively — can be easily solved when both lines are given in parametric form

$$\mathbf{l}_a(s) = \mathbf{m}_a + s \cdot \mathbf{d}_a,$$

$$\mathbf{l}_b(t) = \mathbf{m}_b + t \cdot \mathbf{d}_b. \tag{27}$$

The vector between any two points on the two lines can then be expressed by

$$\mathbf{k}(s,t) = \mathbf{l}_b(t) - \mathbf{l}_a(s) = \mathbf{m}_{ab} + t \cdot \mathbf{d}_b - s \cdot \mathbf{d}_a, \tag{28}$$

where $\mathbf{m}_{ab} = \mathbf{m}_b - \mathbf{m}_a$. The vector $\mathbf{k}(s_0,t_0)$ denotes the vector between the two closest points $\mathbf{l}_{a,0} = \mathbf{l}_a(s_0)$ and $\mathbf{l}_{b,0} = \mathbf{l}_b(t_0)$ and is perpendicular to both, $\mathbf{d}_a$ and $\mathbf{d}_b$.
This defines a system of two linear equations:

$$\mathbf{k}(s_0,t_0) \cdot \mathbf{d}_a = \mathbf{m}_{ab} \cdot \mathbf{d}_a + t_0 \cdot \mathbf{d}_a \cdot \mathbf{d}_b - s_0 = 0 \tag{29}$$

$$\mathbf{k}(s_0,t_0) \cdot \mathbf{d}_b = \mathbf{m}_{ab} \cdot \mathbf{d}_b + t_0 - s_0 \cdot \mathbf{d}_a \cdot \mathbf{d}_b = 0 \tag{30}$$

Solving (29) and (30) for s_0 and t_0 yields

$$s_0 = \frac{(\mathbf{m}_{ab} \cdot \mathbf{d}_a) - (\mathbf{m}_{ab} \cdot \mathbf{d}_b)(\mathbf{d}_a \cdot \mathbf{d}_b)}{1 - (\mathbf{d}_a \cdot \mathbf{d}_b)^2}, \tag{31}$$

and, respectively,

$$t_0 = -\frac{(\mathbf{m}_{ab} \cdot \mathbf{d}_b) - (\mathbf{m}_{ab} \cdot \mathbf{d}_a)(\mathbf{d}_a \cdot \mathbf{d}_b)}{1 - (\mathbf{d}_a \cdot \mathbf{d}_b)^2}. \tag{32}$$

A.3.2 Line-Plane Intersection

An arbitrary plane is defined as the set of points $\mathbf{x}$ that comply with

$$\mathbf{n}(\mathbf{x} - \mathbf{p}) = 0, \tag{33}$$

where $\mathbf{n} = (n_x, n_y, n_z)$ describes the normal vector of the plane and $\mathbf{p}$ a freely chosen reference point that lies within the plane. Given a line with $\mathbf{l}(u) = \mathbf{m} + u \cdot \mathbf{d}$, the solution for the intersection can be found by inserting this into Eq. (33) and solving for u. This results in

$$u = \frac{\mathbf{n}(\mathbf{p} - \mathbf{m})}{\mathbf{n} \cdot \mathbf{d}}. \tag{34}$$

A.3.3 Line-Cylinder Intersection

The calculation of the intersection of a line with an arbitrarily oriented cylinder is shown here after transforming the line into the cartesian frame of the cylinder, i.e. the coordinate system where the symmetry axis of the cylinder builds the z-axis; In this frame the cylinder is centered around the origin and can be projected to a circle in the $x - y$ plane.

Let $\mathbf{p}_1 = (p_{1x}, p_{1y}, p_{1z})$ and $\mathbf{p}_2 = (p_{2x}, p_{2y}, p_{2z})$ be two points of the line in the cylinder frame. The solution can then be found in the projective $x - y$ plane where the lines can be described as $y = kx + d$, with

$$k = \frac{p_{2y} - p_{1y}}{p_{2x} - p_{1x}}, \tag{35}$$

and respectively

$$d = \frac{p_{2x}p_{1y} - p_{1x}p_{2y}}{p_{2x} - p_{1x}}. \tag{36}$$

It intersects the corresponding circle $x^2 + y^2 = R^2$, which can then be found by a simple quadratic equation and reinsertion into the line equation.

A.4 Intersection Formulas for helical Propagation

All calculations performed to find the intersection of the helix with a given `Surface` object are performed in the so-called *helix frame* in which the helix progresses along the z axis and has an initial starting position of $\mathbf{h}_0 = (R, 0, 0)$. The helix parameterisation in this frame is done in the azimuthal angle ϕ and can be expressed as

$$\mathbf{h}(\phi) = \begin{pmatrix} R \cdot \cos\phi \\ R \cdot \sin\phi \\ \hat{o} \cdot R \cdot \phi \cot\theta \end{pmatrix}, \tag{37}$$

when $\hat{o} = \pm 1$ defining the orientation of the helix. The solution for the intersections with different surface types are found by determining the parameter ϕ_i at the intersection point. The momentum vector at the intersection is then expressed as

$$\mathbf{p}(\phi_i) = p \cdot \frac{\mathbf{h}'(\phi_i)}{||\mathbf{h}'(\phi_i)||}, \tag{38}$$

when $\mathbf{h}'(\phi) = \frac{\partial \mathbf{h}(\phi)}{\partial \phi}$.
In general, more than one solution exists for the intersection of a helix with different surfaces. In the ATLAS track reconstruction, however, the helix radius given by the momentum range of particles that are subject of track reconstruction is big in comparison to the detector component to be intersected, thus the solution is in most cases unambiguous.

A.4.1 Helix-Plane Intersection

The intersection of a helix with an plane that is expressed in the helix frame can be found by inserting the helix parameterisation given in Eq. (37) into to the normal vector form of a plane given by Eq. (33). After some manipulations, the intersection solution is given by the root of the function

$$\begin{aligned} f(\phi) &= n_x \cos\phi + n_y \sin\phi + \hat{o} \cdot n_z \cdot \phi \cot\theta - \tfrac{\delta}{R} \\ &= a_1 \cos\phi + a_2 \sin\phi + a_3 \cdot \phi + a_4 \qquad = 0 \end{aligned}, \tag{39}$$

when $\delta = \mathbf{n} \cdot \mathbf{p}$. Equation (39) is characterised by a linear part in ϕ overlaid by an oscillating part with the coefficients a_1 and a_2. Two special cases exist for which the intersection can be calculated analytically: for surfaces parallel to the $x - y$ plane of the helix frame $n_x = n_y = 0$ and the solution is given by the linear equation in ϕ. On the other hand, if the the normal vector of the plane is perpendicular to the z axis of the helix frame, the problem can be solved in the $x - y$ coordinate plane and reduces to the same case as described in Sec. A.3.3. In the general case, however, the solution can not be found analytically it has to be calculated numerically using an iterative approach. In the implementation of the ATLAS `HelixPropagator` a standard Newton-Rhapson method is used for finding the root of Eq. (39). First a starting point for the iterative procedure is found by solving the linear part, which yields

$$\phi_0 = -\frac{a_4}{a_3}. \tag{40}$$

The solution of the azimuthal angle ϕ_i at the intersection point is then found by iterating

$$\phi_i = \phi_{i-1} - \frac{f(\phi_{i-1})}{f'(\phi_{i-1})} \tag{41}$$

when $f'(\phi) = \frac{\partial f(\phi)}{\partial \phi}$. The iteration is stopped once the function value $f(\phi)$ drops below a certain numerical cut value.

A.4.2 Helix-Cylinder Intersection

The intersection of a helix with a cylinder is trivial in the case when the symmetry axis of the cylinder is parallel to the helix guiding center (z axis in the helix frame). It reduces to finding the intersection of two circles in the projected $x - y$ plane and can be solved accordingly. This is the most common case for the `HelixPropagator` since the solenoidal magnetic field in the ATLAS Inner Detector points along the global z axis (and one could assume it to be constant in an defined inner volume) and most cylinders that represent layers or volume boundaries are aligned in the same way.

For completeness, the `HelixPropagator` does also include the calculation with arbitrarily oriented cylinders, which is similar to the intersection of a helix with a plane solved in an iterative Newton-Rhapson approach. In the helix frame, the set of point $\mathbf{x}$ that define a cylinder surface of radius r are given by satisfying the equation

$$|[\mathbf{x} - \mathbf{c}] \times \mathbf{l}_z|^2 = r^2, \tag{42}$$

when $\mathbf{c}$ denotes the center position and $\mathbf{l}_z$ the symmetry axis of the cylinder. Inserting the helix parameterisation $\mathbf{h}(\phi)$ given in Eq. (37) and defining $\mathbf{d} = \mathbf{c} \times \mathbf{l}_z$ follows

$$[\mathbf{h}(\phi) \times \mathbf{l}_z]^2 - 2 \cdot (\gamma(\phi) \times \mathbf{l}_z) \cdot \mathbf{d} + \mathbf{d}^2 - r^2 = 0. \tag{43}$$

Solving Eq. (43) in components yields

$$\begin{aligned} f(\phi) \quad &= a_0 + a_1 \cdot \phi + a_2 \cdot \phi^2 \\ &+ a_3 \cos\phi + a_4 \sin\phi + a_5 \sin(2\phi) \\ &+ a_6 \phi \cos\phi + a_7 \phi \sin\phi \\ &+ a_8 \cos^2\phi + a_9 \sin^2\phi \qquad = 0 \end{aligned} \tag{44}$$

The solution for this equation is again performed in an iterative way using the solution of the quadratic equation in the coefficients a_0, a_1 and a_2 as seed for the iterative solution of the entire expression.

A.4.3 Helix-Line Approach

The calculation of the closest approach of a helix to a line that is parallel to the z axis is trivial, since it is positioned on the plane that is defined through the two parallel lines. For calculating the closest approach of a helix to an arbitrary straight line it is convenient to transform the line into the helix frame representation. The line parameterisation as given in Eq. (27) and the helix expression — Eq. (37) — are used in this frame, respectively. The two points of closest approach $\mathbf{l}_c = \mathbf{l}(t_c)$ on the line, and respectively $\mathbf{h}_c = \mathbf{h}(\phi_c)$ on the helix are then represented by a parameter set (t_c, ϕ_c).

The vector joining these two points $\mathbf{j} = (\mathbf{h}_c - \mathbf{l}_c)$ is orthogonal to both, the line direction and the track direction at $\mathbf{h}_c$ which is expressed through

$$\begin{aligned} \mathbf{j} \cdot \mathbf{d} \quad &= \quad 0 \\ \mathbf{j} \cdot \tfrac{\partial \mathbf{h}_c}{\partial \phi} \quad &= \quad 0 \end{aligned} \tag{45}$$

Solving the first equation for t_c and inserting the solution into latter, yields, after doing some manipulation,

$$\begin{aligned} f(\phi) \quad &= a_0 + a_1\phi \\ &+ a_2 \cdot sin(\phi) + a_3 \cdot cos(\phi) + a_4 \cdot sin(2\phi) + a_5 \cdot cos(2\phi) \\ &+ a_6 \cdot \phi \cdot sin(\phi) + a_7 \cdot \phi \cdot cos(\phi) \qquad = 0 \end{aligned} \tag{46}$$

The solution is then carried out in the same way as for finding the intersection of a line with a cylinder.

A.5 Analytic Transport of the Covariance Matrix using the curvilinear Frame

The analytic transport of the covariance matrix along the the curvilinear frame is shown for a helical track model and follows the path laid out in [11], but is adapted for the ATLAS track parameterisation. Since the straight line propagation is just one specific type of a helix propagation it can be evidently applied to this case as well. The initial problem of transporting a local covariance matrix $\mathbf{C}_{li}$ from an initial surface to a target surface is hereby split into the transformations carried out between the local and curvilinear frame on the starting surface, the transport of the covariance along the curvilinear track frame and the final transformation between the curvilinear frame at the destination surface and its attached measurement frame, see Eq. (4). The curvilinear frame is hereby defined as given in Eq. (2).

A helical trajectory of a particle with charge q and momentum magnitude p in a constant magnetic field $\mathbf{B}$ can be parameterised as the global position $\mathbf{m}(\psi)$ and direction $\mathbf{t}(\psi)$ in a three dimensional space

$$\mathbf{m}(\psi) = \mathbf{m}_0 + \frac{\gamma}{Q}(\psi - \sin\psi)\mathbf{h} + \frac{\sin\psi}{Q}\mathbf{t}_0 + \frac{\alpha}{Q}(1-\cos\psi)\mathbf{n}_0$$

$$\mathbf{t}(\psi) = \frac{\partial \mathbf{m}(\psi)}{\partial s} = \frac{\partial \mathbf{m}(\psi)}{\partial \psi}\cdot\frac{\partial \psi}{\partial s} = \gamma(1-\cos\psi)\mathbf{h} + \cos\psi\cdot\mathbf{t}_0 + \alpha\sin\psi\cdot\mathbf{n}_0 \tag{47}$$

when $\psi = Q\cdot s$ denotes the momentum scaled run parameter s, $Q = -|\mathbf{B}|\frac{q}{p}$, and the initial vectors at $s=0$ are written as $\mathbf{m}_0$ and $\mathbf{t}_0$, respectively. The vector $\mathbf{h}$ describes the direction of the magnetic field, $\mathbf{n} = \frac{\mathbf{h}\times\mathbf{t}}{\alpha}$ the normalised *trihedron* vector, α evidently the length of $\mathbf{h}\times\mathbf{t}$ and γ the projection of $\mathbf{h}$ onto $\mathbf{t}$.

We now investigate the influence of small variations on the start parameters $\mathbf{m}_0$, $\mathbf{t}_0$ and $\frac{1}{p_0}$ on the transported parameters $\mathbf{m}$, $\mathbf{t}$. The variations are applied through a displacement by $d\mathbf{m}_0$ that is restricted to the curvilinear $u-v$ plane, a deflection $d\mathbf{t}_0$ to the direction and a modification of the curvature given by $\delta(q/p_0)$. The variations result not only in modified target parameters but also in a variation of propagation length s to comply with the orthogonality relation

$$d\mathbf{m}\cdot\mathbf{t} = 0. \tag{48}$$

The total differentials $d\mathbf{m}$ and $d\mathbf{t}$ that incorporate the resulting variations on the the target surface with respect to modified starting parameters are build as

$$d\mathbf{m} = \frac{\partial\mathbf{m}}{\partial\mathbf{m}_0}d\mathbf{m}_0{}^{(a)} + \frac{\partial\mathbf{m}}{\partial\mathbf{t}_0}d\mathbf{t}_0{}^{(b)} + \frac{\partial\mathbf{m}}{\partial(1/p_0)}\delta(1/p_0)^{(c)} + \frac{\partial\mathbf{m}}{\partial s}\delta s^{(d)} \tag{49}$$

and, respectively,

$$d\mathbf{t} = \frac{\partial\mathbf{t}}{\partial\mathbf{t}_0}d\mathbf{t}_0{}^{(e)} + \frac{\partial\mathbf{t}}{\partial(1/p_0)}\delta(1/p_0)^{(f)} + \frac{\partial\mathbf{t}}{\partial s}\delta s^{(g)}. \tag{50}$$

Figure 20 shows the fundamental relations between the applied variations and their effects on the target surface.

Curvilinear to Curvilinear Transformations In the following, the transport of the covariance matrix between to curvilinear frames is shown to illustrate the principle of the further calculations.

We can, by inserting the partial derivatives of Eq. (47) identify the components of Eq. (49) and Eq. (50) as

$$\frac{\partial\mathbf{m}}{\partial\mathbf{m}_0}d\mathbf{m}_0 = d\mathbf{m}_0 \tag{51a}$$

$$\frac{\partial\mathbf{m}}{\partial\mathbf{t}_0}d\mathbf{t}_0 = \frac{\psi-\sin\psi}{Q}\mathbf{h}(\mathbf{h}\cdot d\mathbf{t}_0) + \frac{\sin\psi}{Q}d\mathbf{t}_0 + \frac{1-\cos\psi}{Q}(\mathbf{h}\times d\mathbf{t}_0) \tag{51b}$$

$$\frac{\partial\mathbf{m}}{\partial(1/p_0)}\delta(1/p_0) = p[s\cdot\mathbf{t} + \mathbf{m}_0 - \mathbf{m}]\cdot\delta(1/p_0) \tag{51c}$$

$$\frac{\partial\mathbf{m}}{\partial s}\delta s = \mathbf{t}\cdot\delta s \tag{51d}$$

$$\frac{\partial\mathbf{t}}{\partial\mathbf{t}_0}d\mathbf{t}_0 = (1-\cos\psi)\mathbf{h}(\mathbf{h}\cdot d\mathbf{t}_0) + \cos\psi + \sin\psi(\mathbf{h}\times d\mathbf{t}_0) \tag{51e}$$

$$\frac{\partial\mathbf{t}}{\partial(1/p_0)}\delta(1/p_0) = \alpha\cdot Q\cdot s\cdot p\cdot\mathbf{n}\cdot\delta(1/p_0) \tag{51f}$$

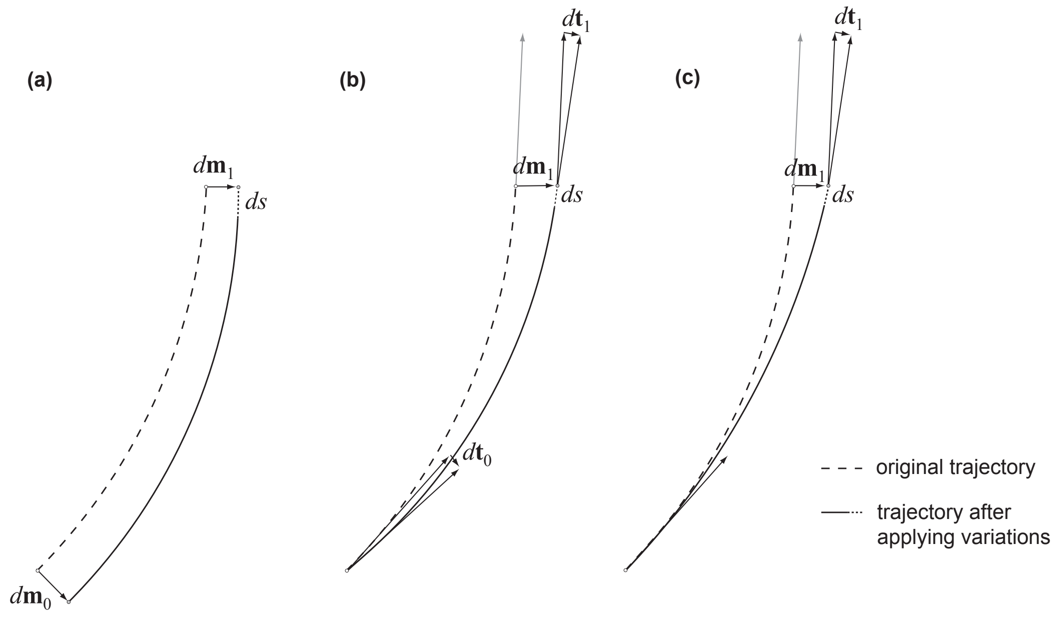


Figure 20: The initial variations on the track state and the directly resulting variations at the target position of the extrapolation process. The illustration decouples the possible start parameter variations: **(a)** illustrates the effect of an initial variation $d\mathbf{m}_0$ at the starting position, while **(b)** shows the effect of a slightly modified momentum direction; **(c)** displays the changed trajectory for a modified curvature while keeping the same initial direction.

$$\frac{\partial \mathbf{t}}{\partial s}\delta s = \alpha \cdot Q \cdot \mathbf{n} \cdot \delta s \tag{51g}$$

We used that $\gamma = \mathbf{h}\cdot\mathbf{t} = \mathbf{h}\cdot\mathbf{t}_0$ remains constant along s and so does α. Let us in addition mention that Eq. (51f) follows directly from Eq. (51c) and that Eq. (51g) represents the centripetal acceleration caused by the Lorentz force.

We search for transport jacobian matrix for the errors on the initial curvilinear parameters $\mathbf{c}_i = (u_i, v_i, \phi_i\,, \theta_i\,, 1/p_i)^T$ and the final curvilinear parameters $\mathbf{c}_f = (u_f, v_f, \phi_f\,, \theta, 1/p_f)^T$. In other words, we look for the coefficients C_{if} that transform a variation of the initial parameter parameter set c_i to the final parameters set $\mathbf{c}_f$. As an example we will specify these components for the variations on δu_f:

$$\delta u_f = \frac{\partial u_f}{\partial u_i}\delta u_i + \frac{\partial u_f}{\partial v_i}\delta v_i + \frac{\partial u_f}{\partial \phi_i}\delta\phi_i + \frac{\partial u_f}{\partial \theta_i}\delta\theta_i + \frac{\partial u_f}{\partial(1/p_0)}\delta(1/p_0). \tag{52}$$

The track parameters q/p as defined through the ATALS tracking EDM is hereby replaced by the $1/p$ since the variations on the signed momentum representation is restricted to change the momentum magnitude only, while leaving the charge parameter untouched. In any curvilinear frame, the total differentials $d\mathbf{m}$ and $d\mathbf{t}$ can be expressed with respect to the curvilinear coordinates and are then given by[19]

$$d\mathbf{m} = \mathbf{e}_u\,\delta u + \mathbf{e}_v\,\delta v \tag{53}$$

$$d\mathbf{t} = \frac{\partial \mathbf{t}}{\partial \phi}\delta\phi + \frac{\partial \mathbf{t}}{\partial \theta}\delta\theta = \sin\theta \mathbf{e}_u\,\delta\phi - \mathbf{e}_v\,\delta\theta \tag{54}$$

Equation (54) can be shown when differentiating $\mathbf{t}$ with respect to ϕ and θ and using the fact that the curvilinear frame is constructed by using the global z axis.

Without losing generality, we will in the following use Eqs. (47) to (51) assuming that the initial frame is given at $s = 0$, i.e. $\mathbf{m}_i = \mathbf{m}_0$ and denote $\psi_{if} = \psi_f - \psi_i \equiv \psi$ and $s_{if} \equiv s$ for convenience.

[19]Note that the orthogonality relation as required by Eq. (48) is hereby met by restricting the total differential to a variation in $\mathbf{e}_u$ and $\mathbf{e}_v$, respectively.

After inserting Eqs. (53) and (54) into Eq. (49) and using the partial derivatives found in Eq. (51) we gain an expression that correlates the variations on the local coordinates in the final curvilinear frame with the variations of the start parameters

$$\begin{aligned}
\mathbf{e}_u^f \cdot \delta u_f + \mathbf{e}_v^f \cdot \delta v_f &= \mathbf{e}_u^i \cdot \delta u_i \\
&+ \mathbf{e}_v^i \cdot \delta v_i \\
&+ \tfrac{\sin\theta_i}{Q}\left[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)\mathbf{h} + \sin\psi\mathbf{e}_u^i + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_u^i)\right]\cdot\delta\phi_i \\
&- \tfrac{1}{Q}\left[\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)\mathbf{h} + + \sin\psi\mathbf{e}_v^i + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_v^i)\right]\cdot\delta\theta_i \\
&+ p\cdot[s\cdot\mathbf{t}_f + \mathbf{m}_i - \mathbf{m}_f]\cdot\delta(1/p_i) \\
&+ \mathbf{t}\cdot\delta s.
\end{aligned} \tag{55}$$

This equation is multiplied by $\mathbf{e}_u^f$ to isolate the terms given in Eq. (52), since it cancels the contributions in the orthogonal component $\mathbf{t}_f \equiv \mathbf{e}_t^f$. This yields

$$\begin{aligned}
\delta u_f &= \mathbf{e}_u^i \cdot \mathbf{e}_u^f\,\delta u_i \\
&+ \mathbf{e}_v^i \cdot \mathbf{e}_u^f\,\delta v_i \\
&+ \tfrac{\sin\theta_i}{Q}\left[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)(\mathbf{h}\cdot\mathbf{e}_u^f) + \sin\psi(\mathbf{e}_u^i\cdot\mathbf{e}_u^f) + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_u^i)\right]\cdot\delta\phi_i \\
&- \tfrac{1}{Q}\left[\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)(\mathbf{h}\cdot\mathbf{e}_u^f) + + \sin\psi(\mathbf{e}_v^i\cdot\mathbf{e}_u^f) + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{e}_u^f\right]\cdot\delta\theta_i \\
&+ p\cdot\left[(\mathbf{m}_i\cdot\mathbf{e}_u^f) - (\mathbf{m}_f\cdot\mathbf{e}_u^f)\right]\cdot\delta(1/p_i).
\end{aligned} \tag{56}$$

By comparing the coefficients of Eq. (56) with those defined in Eq. (52), the first row of the Jacobian matrix is found and since Eq. (55) is on the left hand side symmetrical in $\mathbf{e}_u$ and $\mathbf{e}_v$, a similar equation for δv_f can be obtained. For evaluating the covariance entries that regulate the transport of the directional uncertainties we take Eq. (50) and insert again both, the partial derivatives as given in Eq. (51) and the expression of the total differential in the curvilinear frame coordinates. This yields the expression

$$\begin{aligned}
\sin\theta_f\,\mathbf{e}_u^f\,\delta\phi_f - \mathbf{e}_v^f\,\delta\theta_f &= \\
&+ \sin\theta_i\left[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)\mathbf{h} + \cos\psi\mathbf{e}_u^i + \sin\psi(\mathbf{h}\times\mathbf{e}_u^i)\right]\delta\phi_i \\
&- \left[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)\mathbf{h} + \cos\psi\mathbf{e}_v^i + \sin\psi(\mathbf{h}\times\mathbf{e}_v^i)\right]\delta\theta_i \\
&+ \alpha\cdot Q\cdot p\cdot s\delta(1/p_i) \\
&+ \alpha\cdot Q\cdot\mathbf{n}\cdot\delta s
\end{aligned} \tag{57}$$

We first try to eliminate the explicit variation on the propagation length δs from Eq. (57) before we can proceed identically as for the local coordinates. Taking Eq. (49) and multiply $d\mathbf{m}_f$ with the perpendicular direction $\mathbf{t}_f$ gives an expression for δs as

$$\delta s = -d\mathbf{m}_i\cdot\mathbf{t}_f - \left[\frac{\partial\mathbf{m}_f}{\partial\mathbf{t}_i}d\mathbf{t}_i\right]\cdot\mathbf{t}_f - \left[\frac{\partial\mathbf{m}_f}{\partial(1/p_i)}\cdot\mathbf{t}_f\right]\cdot\delta(1/p_i), \tag{58}$$

which we insert in Eq. (57). When multiplying Eq. (58) again with $\mathbf{e}_u^f$ and $\mathbf{e}_v^f$ and and isolating the components, the third and fourth row — describing the propagation of the directional uncertainties — of the transport Jacobian is determined. The missing terms of the Jacobian matrix can be found without additional calculations; they are

$$\frac{\partial(1/p_f)}{\partial(1/p_i)} = 1 \tag{59}$$

and, respectively,

$$\frac{\partial(1/p_f)}{\partial(u_i, v_i, \phi_i, \theta_i)} = 0 \tag{60}$$

and correspond to momentum conservation and an interaction-free transport.

The fully deployed Jacobian matrix describing the transport from one curvilinear frame to another can be found in Fig. 21.

$$\mathbf{J}_{ci,cf} =$$

$$\left(\begin{array}{c|c|c|c|c}
\mathbf{e}^f_u \cdot \mathbf{e}^i_u & \mathbf{e}^f_u \cdot \mathbf{e}^i_v & \begin{array}{c} -\frac{\sin\theta_i}{Q}[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}^i_u)(\mathbf{h}\cdot\mathbf{e}^f_u) \\ +\sin\psi(\mathbf{e}^i_u \cdot \mathbf{e}^f_u) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}^i_u)\mathbf{e}^f_u)] \end{array} & \begin{array}{c} \frac{1}{Q}[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}^i_v)(\mathbf{h}\cdot\mathbf{e}^f_u) \\ +\sin\psi(\mathbf{e}^i_v \cdot \mathbf{e}^f_u) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}^i_v)\mathbf{e}^f_u)] \end{array} & p\cdot(\mathbf{m}_i - \mathbf{m}_f)\mathbf{e}^f_u \\
\hline
\mathbf{e}^f_v \cdot \mathbf{e}^i_u & \mathbf{e}^f_v \cdot \mathbf{e}^i_v & \begin{array}{c} -\frac{\sin\theta_i}{Q}[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}^i_u)(\mathbf{h}\cdot\mathbf{e}^f_v) \\ +\sin\psi(\mathbf{e}^i_u \cdot \mathbf{e}^f_v) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}^i_u)\mathbf{e}^f_v)] \end{array} & \begin{array}{c} \frac{1}{Q}[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}^i_v)(\mathbf{h}\cdot\mathbf{e}^f_v) \\ +\sin\psi(\mathbf{e}^i_v \cdot \mathbf{e}^f_v) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}^i_v)\mathbf{e}^f_v)] \end{array} & p\cdot(\mathbf{m}_i - \mathbf{m}_f)\mathbf{e}^f_v \\
\hline
\frac{\alpha\cdot Q}{\sin\theta_f}(\mathbf{n}\cdot\mathbf{e}^f_u)(\mathbf{e}^i_u \cdot \mathbf{t}_f) & \frac{\alpha\cdot Q}{\sin\theta_f}(\mathbf{n}\cdot\mathbf{e}^f_u)(\mathbf{e}^i_v \cdot \mathbf{t}_f) & \begin{array}{c} \frac{\sin\theta_i}{\sin\theta_f}[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}^i_u)(\mathbf{h}\cdot\mathbf{e}^f_u) \\ +\cos\psi(\mathbf{e}^i_u \cdot \mathbf{e}^f_u) + \sin\psi(\mathbf{h}\times\mathbf{e}^i_u)\mathbf{e}^f_u \\ -\alpha(\mathbf{n}\cdot\mathbf{e}^f_u)\{(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}^i_u) \\ +\sin\psi(\mathbf{e}^i_u \cdot \mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}^i_u)\mathbf{t}_f\}] \end{array} & \begin{array}{c} -\frac{1}{\sin\theta_f}[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}^i_v)(\mathbf{h}\cdot\mathbf{e}^f_u) \\ +\cos\psi(\mathbf{e}^i_v \cdot \mathbf{e}^f_u) + \sin\psi(\mathbf{h}\times\mathbf{e}^i_v)\mathbf{e}^f_u \\ -\alpha(\mathbf{n}\cdot\mathbf{e}^f_u)\{(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}^i_v) \\ +\sin\psi(\mathbf{e}^i_v \cdot \mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}^i_v)\mathbf{t}_f\}] \end{array} & \begin{array}{c} \frac{\alpha\cdot p\cdot Q}{\sin\theta_f}(\mathbf{n}\cdot\mathbf{e}^f_u) \\ \cdot[\mathbf{t}_f(\mathbf{m}_i - \mathbf{m}_f)] \end{array} \\
\hline
-\alpha\cdot Q(\mathbf{n}\cdot\mathbf{e}^f_v)(\mathbf{e}^i_u \cdot \mathbf{t}_f) & -\alpha\cdot Q(\mathbf{n}\cdot\mathbf{e}^f_v)(\mathbf{e}^i_v \cdot \mathbf{t}_f) & \begin{array}{c} -\sin\theta_i[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}^i_u)(\mathbf{h}\cdot\mathbf{e}^f_v) \\ +\cos\psi(\mathbf{e}^i_u \cdot \mathbf{e}^f_v) + \sin\psi(\mathbf{h}\times\mathbf{e}^i_u)\mathbf{e}^f_v \\ -\alpha(\mathbf{n}\cdot\mathbf{e}^f_v)\{(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}^i_u) \\ +\sin\psi(\mathbf{e}^i_u \cdot \mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}^i_u)\mathbf{t}_f\}] \end{array} & \begin{array}{c} (1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}^i_v)(\mathbf{h}\cdot\mathbf{e}^f_v) \\ +\cos\psi(\mathbf{e}^i_v \cdot \mathbf{e}^f_v) + \sin\psi(\mathbf{h}\times\mathbf{e}^i_v)\mathbf{e}^f_v \\ -\alpha(\mathbf{n}\cdot\mathbf{e}^f_v)\{(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}^i_v) \\ +\sin\psi(\mathbf{e}^i_v \cdot \mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}^i_v)\mathbf{t}_f\} \end{array} & \begin{array}{c} -\alpha\cdot p\cdot Q(\mathbf{n}\cdot\mathbf{e}^f_v) \\ \cdot[\mathbf{t}_f(\mathbf{m}_i - \mathbf{m}_f)] \end{array} \\
\hline
0 & 0 & 0 & 0 & 1
\end{array}\right)$$

Figure 21: The full deployed Jacobian matrix for the transportation of errors on the track parameters between two curvilinear frames on a track for a helical track model. For a straight line track model, the Jacobian matrix reduces quite substantially.

Curvilinear to Local Transformations and vice versa For a full transport of the covariance matrix between to arbitrary track representations it is also necessary to determine the Jacobian matrices that transform the local parameter expressions $\mathbf{x} = (l_1, l_2, \phi, \theta, q/p)^T$ to the curvilinear parameters $\mathbf{c} = (u, v, \phi, \theta, 1/p)$ and the reverse way. In ATLAS, all track representations can be expressed on a planar surface, and including an additional transformation, even in a two-dimensional cartesian coordinate system, which thus remains the only case to be investigated in this scope (this is guaranteed through the *measurement frame* mechanism, see [6]).

We use the definition of a lateral variation in the curvilinear frame — Eq. (53) — and express it through a variation in the local coordinates and along the momentum

$$d\mathbf{m} = \mathbf{e}_u\,\delta u + \mathbf{e}_v\,\delta v = \mathbf{e}_1\delta l_1 + \mathbf{e}_2\delta l_2 + \mathbf{t}\cdot\delta s, \tag{61}$$

and redo the same for the variation on the momentum

$$d\mathbf{t} = \sin\theta\mathbf{e}_u\,\delta\phi - \mathbf{e}_v\,\delta\theta = \sin\theta\mathbf{e}_u\,\delta\phi - \mathbf{e}_v\,\delta\theta + \frac{\partial\mathbf{t}}{\partial s}\delta s. \tag{62}$$

It is worth mentioning that although the directional expression is identical in both local and curvilinear frame, the angular parameters ϕ, θ have to be first treated independently since we want to investigate both sides individually.

Multiplying Eq. (61) with the orthogonal vector $\mathbf{t}$ (which eliminates components in $\mathbf{e}_u$ and $\mathbf{e}_v$) and solving for δs establishes an expression of the necessary[20] variation in the propagation length caused by a local variation on the starting surface

$$\delta s = -(\mathbf{t}\cdot\mathbf{e}_1)\delta l_1 - (\mathbf{t}\cdot\mathbf{e}_2)\delta l_2, \tag{63}$$

while the multiplication with $\mathbf{e}_u$ and $\mathbf{e}_v$, respectively, yields the expression of the variations in δu and δv

$$\begin{aligned}\delta u &= (\mathbf{e}_u\cdot\mathbf{e}_1)\delta l_1 + (\mathbf{e}_u\cdot\mathbf{e}_2)\delta l_2\\ \delta v &= (\mathbf{e}_v\cdot\mathbf{e}_1)\delta l_1 + (\mathbf{e}_v\cdot\mathbf{e}_2)\delta l_2.\end{aligned} \tag{64}$$

The coefficients of the Jacobian matrix that relate the local surface parameters with the curvilinear parameters u and v can be directly gained from Eq. (64). In a second step, Eq. (62), is multiplied by $\mathbf{e}_u$ and $\mathbf{e}_v$ and Eq. (63) is inserted to eliminate δs. This yields the additional components that relate the angular variables with the local surface parameters.

The reverse transformation from the (transported) curvilinear frame to the local frame of the target surface can be found in a similar way, starting from Eq. (61). We clearly want to evaluate the variations hereby in the target frame, i.e. we demand that $d\mathbf{m}$ is orthogonal to $\mathbf{e}_3$ in this case, which builds the normal vector of the target surface. Under this assumption, the multiplication of Eq. (61) with $\mathbf{e}_3$ yields

$$\delta s = \frac{\mathbf{e}_u\cdot\mathbf{e}_3}{\mathbf{t}\cdot\mathbf{e}_3}\delta u + \frac{\mathbf{e}_v\cdot\mathbf{e}_3}{\mathbf{t}\cdot\mathbf{e}_3}\delta v. \tag{65}$$

Introducing Eq. (65) to Eq. (61) now enhances the same procedure to find the components for the inverse Jacobian matrix. The fully deployed matrices can be found in the code documentation of the TrkExUtils CVS package [30].

References

[1] R. Frühwirth et al., *Application of Kalman Filtering to Track and Vertex Fitting*, Nucl. Inst. Meth., **A 262**, 1987.

[2] V. Boisvert et al, *Final Report of the ATLAS Reconstruction Task Force (RTF)*, ATLAS Note, ATL-SOFT-2003-01, 2003.

[3] Athena homepage, *http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/*.

[20] To comply with the orthogonality $d\mathbf{m}\cdot\mathbf{t} = 0$.

[4] ATLAS Quality Assurance Group, *Atlas C++ Coding Standard Specifications*, ATLAS Note, ATL-SOFT-2002-001, 2002.

[5] Atlas Collaboration, *ATLAS Computing Technical Design Report*, ATLAS TDR, CERN-LHCC-2005-022, 2005.

[6] F. Akesson et al., *The ATLAS Tracking Event Data Model*, ATLAS Public Note, ATL-SOFT-PUB-2006-004, 2006.

[7] A. Salzburger, S. Todorova and M. Wolter, *The Atlas Tracking Geometry Description*, ATLAS communication to be published, ATL-COM-SOFT-2007-009, 2007.

[8] T. Cornelissen et al., *Updates of the ATLAS Tracking Event Data Model*, ATLAS communication to be published, ATL-COM-SOFT-2007-008, 2007.

[9] J.R. Dormand and P.J. Price, *A family of embedded Runge-Kutta formulae*, J. Comp. Appl. Math.,**6**, 1980.

[10] A. Salzburger, *The new Fast ATLAS Track Simulation (FATRAS)*, Proceeding of CHEP 2006, 2006.

[11] A. Strandlie and W. Wittek, *Derivation of Jacobians for the propagation of covariance matrices of track parameters in homogeneous magnetic fields*, Nucl. Inst. & Meth. in Phys., **A 566**, 2006.

[12] E. Lund, L. Bugge, A. Strandlie and I. Gavrilenko, *Simulatneous Track and Error Propagation with Material Effects*, ATLAS Note in preparation

[13] V.L. Highland, *Some Practical Remarks on Multiple Scattering*, Nucl. Inst. & Meth. **129**, 1975, and *Erratum*, Nucl. Inst. & Meth., **161**, 1979.

[14] G. Moliére, *Therorie der Streuung schneller geladener Teilchen I. Einzelstreuung am abgeschirmten Coulomb-Feld*, Z. Naturforschung **2a**, 1947, and *Theorie der Streuung schneller geladener Teilchen II. Mehrfach- und Vielfachstreuung*, Z. Naturforschung **3a**, 1948.

[15] B. Rossi and K. Greisen, *Cosmic-Ray Theory*, Rev. Mod. Phys.,**13**, 1941.

[16] R. Frühwirth, M. Regler, *On the quantitative modeling of core and tails of multiple scattering by Gaussian mixtures*, Nucl. Inst. & Meth. **A 456**, 2001.

[17] R. Frühwirth, M. Liendl, *Mixture models of multiple scattering: computation and simulation*, Comp. Phys. Comm. textbf141, 2001.

[18] Agostinelli et al.,*GEANT4: A Simulation toolkit*, Nucl. Inst. & Meth., **A 506**, 2003.

[19] H. Bethe, *Zur Theorie des Durchgangs schneller Korpuskularstrahlen durch Materie*, Annalen der Physik, **397**, 1930.

[20] Lohmann W., Kopp R. Voss, R., *Energy loss of muons in the energy range 1-10 000 GeV*, CERN-85-03 Yellow Reoprt, 1985.

[21] D.E. Groom, N.V. Mokhov and S.I. Striganov, *Muon stopping power and range tables 10 MeV-100 TeV*, Atomic Data and Nuclear Tables, **78**, 2001.

[22] Particle Data Group, *Review of Particle Physics*, Phys. Rev., **D 66**, 2002.

[23] L.D. Landau, *On the energy loss of fast particles by ionisation*, J. Phys. USSR, **8**, 1944.

[24] H. Bethe and W. Heitler, *On the stopping of fast particles and the creation of positive electrons*, Proc. Roy. Soc.**A 146**, 1934.

[25] T.M. Atkinson, *Electron Reconstruction with the ATLAS Inner Detector*, PhD thesis, University of Melbourne, 2006.

[26] Muonboy homepage, *http://cern.ch/atlas-samusog/muonboy/Muonboy.htm.*

[27] C.J.F. Ridders, *Advances in Engineering Software*, **Vol. 4**, 1982.

[28] T.G. Cornelissen, *Track Fitting in the ATLAS Experiment*, CERN-THESIS-2006-07, 2007.

[29] A. Salzburger (Editor) et. al., *Concepts, Design and Implementation of the ATLAS New Track Reconstruction (NEWT)*, ATLAS Communication to be published, ATLAS-COM-SOFT-2007-002, 2007.

[30] ATLAS software CVS repository, online CVSView, *http://atlas-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/*