THE TRACK EXTRAPOLATION PACKAGE IN THE NEW ATLAS TRACKING REALM

A. Salzburger*, Leopold-Franzens-University, Innsbruck, Austria

Abstract

The extrapolation of track parameters and their associated covariance matrices to arbitrarily oriented surfaces of different types inside a non-uniform magnetic field is a fundamental element of any tracking software. It has to take multiple scattering and energy loss effects along the propagated trajectories into account. A good performance in respect of computing time consumption is crucial due to hit and track multiplicity in high luminosity events at the LHC and the small time window of the ATLAS high level trigger. Therefore stable and fast algorithms for the transport of the track parameters and their associated covariance matrices in specific representations to different surfaces in the detector are required. The recently developed track extrapolation package inside the new ATLAS offline tracking software is presented in this document.

INTRODUCTION

During the recent redesign of the ATLAS offline reconstruction software, a new track extrapolation package has been developed within the C++ based software framework ATHENA [1]. The transportation of track parameters and their associated covariance matrices to a given detector surface is a fundamental and frequently performed process in most track fitting algorithms. In general, the extrapolation process can be divided into two parts. The first step is the geometrical transport of the track parameters respectively covariance matrices to given surfaces and will be in the following referred to as propagation, Fig. 1 shows a simplified illustration of such a propagation. The second procedure is the update of the propagated parameters and errors, taking multiple Coulomb scattering and energy loss effects during the propagation process into account. This note covers mainly the first part of the extrapolation process.

THE EXTRAPOLATION PACKAGE DESIGN

The extrapolation package is fully integrated into the ATHENA framework and based on the recently developed ATLAS Event Data Model (EDM) with its associated common tracking algorithms and data classes [2]. The main ingredients of the extrapolation package are AlgTool classes that inherit from the GAUDI [3] AlgTool interface class. AlgTools are managed by a central GAUDI service and can be retrieved from this service at any point in the program flow.

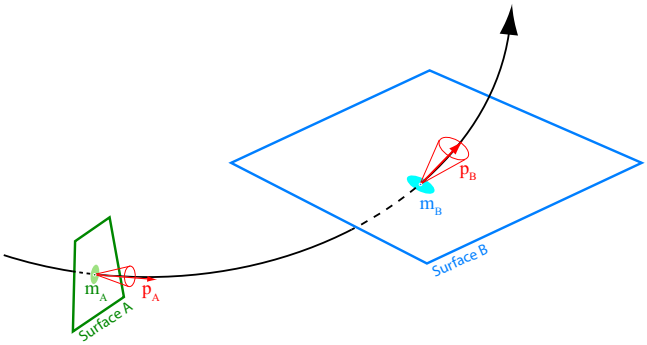

Figure 1: Schematic representation of the transportation of track parameters and their associated errors from Surface A to Surface B. The track parameters on surface i are illustrated by a local position m_i and its error ellipse, such as a momentum vector p_i and a cone representing the error on the momentum direction. The error on the magnitude of the momentum is omitted in this illustration.

The steering of the propagation setup, including the setup of the magnetic field, the propagation algorithm and surface finding logics is done by dedicated python classes.

The extrapolation process can be driven in two different modes, a preconfigured and an unconfigured mode. In the preconfigured mode, the underlying propagation setup (type of propagation, magnetic field setup) is fully determined at startup of the program and should be used for expectable situations in the program flow.

The unconfigured mode is characterized by the fact that the Propagator AlgTool itself is passed to the Extrapolator AlgTool, following a strategy pattern design. This allows an optimization of the extrapolation process by dynamically choosing the propagation algorithms depending on the situation specific parameters, such as the magnetic field, an estimated propagation distance or even starting track parameters characteristics. Various instances of Extrapolator AlgTools in different configurations can be used in parallel.

Design Principles

The following design principles have been respected during the implementation of the ATLAS tracking EDM:

- *Lazy Initialization:* A major design pattern for all data classes in the new ATLAS tracking realm has been the concept of *lazy initialization*, i.e. that quantities (im-

* Andreas.Salzburger@cern.ch

plemented as private member variables) are calculated only when they are first used or requested by a public method. This requires private members to be implemented as mutable pointers, that are at instantiation time of the object initialized to be NULL. Once a private member has been instantiated (by using the `new` operator), it would be cached and not recalculated respectively reinstantiated at the time of a future use.

- *No modifications of instances allowed:* In general, data classes do not allow modifications on their private members from outside once they are instantiated. Explicitly there are no set methods for private members of the class implemented, to avoid conflicts with the concept of *lazy initialization*. A modified object has to be instantiated as a new object.

- *No isValid() methods:* Algorithms and AlgTools return created objects as pointers to new objects. If the performed operation has not been successful, a NULL pointer is returned instead. E.g. a propagation to a surface that can not be hit for geometrical reasons would return a NULL pointer.

The extensive use of the `new` operator might cause an increase of memory allocation time, a future use of the DataPool memory allocation algorithms inside the ATLAS StoreGate Data Model [4] is planned for optimization.

Extrapolation and Propagation AlgTools

During the extrapolation process, the Extrapolator AlgTool acts as the central steering module: it receives the input object of type Track or TrackParameters, together with a destination surface to propagate to. The geometrical propagation is delegated by the Extrapolator AlgTool to the Propagator AlgTool. In addition the Extrapolator AlgTool is responsible for collecting magnetic field and surface information from the appropriate services. The update of the track parameters respectively covariances taking multiple scattering and energy loss into account has to be done after the propagation, the associated tools providing the information to the Extrapolator AlgTool are currently in development.

Surfaces and Track Representations on Surfaces

A new common geometry description usable for tracking algorithms has been developed and integrated in the new tracking software. Surface classes have been designed to handle both, the representation of actual detector elements such as described by the ATLAS detector description GeoModel, and the possibility to instantiate virtual tracking surfaces, such as an imaginary cylinder covering the Inner Detector volume. Five different concrete surface implementations exist respecting surface types with different local frame definitions, see Fig. 3. In case a surface owns a pointer to an existing detector element in the detector store, the entire geometrical information, i.e. the position, the orientation and the boundaries, are directly taken from this source and passed through by the surface class. In this case the surface acts as a light-weighted wrapper around the detector element to interface it with tracking algorithms and data classes.

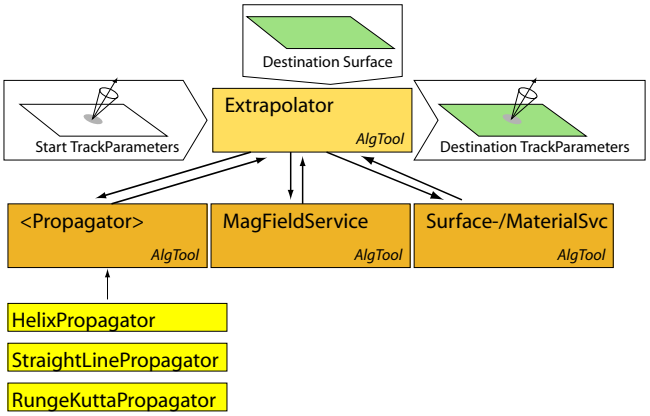

Figure 2: Overview of the extrapolation process: the input is a given start set of track parameters and a destination surface, the output is a pointer to a new instance of type TrackParameters or a NULL pointer in case of an unsuccessful propagation. The propagation AlgTools can be regarded as an input parameter as well in case of the extrapolation AlgTool being driven in an unconfigured setup.

The representation of a track on a specific surface can be done by a set of five track parameters. In the ATLAS EDM these five parameters are the two local coordinates of the surface and a global momentum representation. For each different surface type there is a dedicated type of local track parameters implemented. Respecting the requirements given by the geometry of the ATLAS tracking detectors, only three track parameters types exist in both a measured and an unmeasured flavor. Figure 4 gives an overview of the inheritance structure for the track parameters classes.

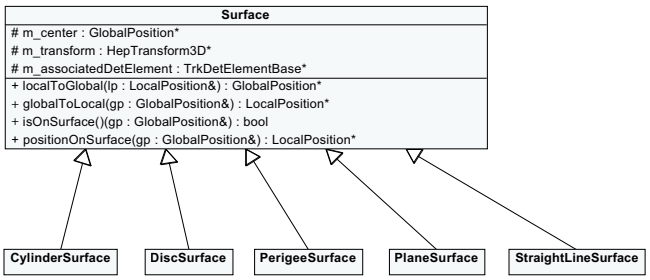

Figure 3: Simplified UML diagram of the tracking detector description in the new ATLAS Event Data Model. All different surface types inherit form a Surface base class and are characterized by a specific local frame. Each child class implements the transformations between local and global coordinates respecting the internal local frame definition.

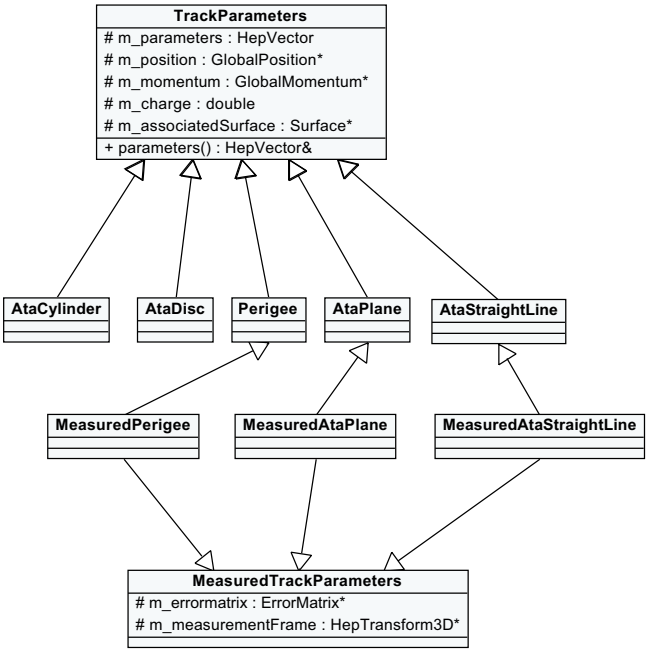

Figure 4: Simplified UML diagram of the track parameter classes in the new ATLAS Event Data Model. For each existing surface type exists an associated TrackParameters class. Measured track parameters exist for PlaneSurface, StraightLineSurface and for the Perigee representation.

THE PROPAGATION PROCESS

The propagation process can be divided into the transport of the five track parameters to the local frame of the destination surface and the propagation of the associated covariance matrix.

Propagation of Track Parameters

The ATLAS track extrapolation package contains three Propagator AlgTools with different underlying track models, see Tab. 1. The propagation of track parameters to any arbitrarily oriented surface can be solved fully analytically only in case of a straight line track model. For a helical track model, the propagation of track parameters to an arbitrarily oriented planar, cylindrical or line-like surface requires an iterative approach. In the current implementation about three to four iteration steps have to be taken to find a converged solution within the required precision.

The different track models are implemented as a line and a helix class inheriting from a common base class. These classes are used by the StraightLinePropagator respectively the HelixPropagator. The RungeKuttaPropagator uses an internal representation and approximates the propagation by step-wise helical track segments.

Propagation of Covariance Matrices

Assuming a linear or a helical track model the propagation of the covariance matrice to a given surface can be done analytically. The errors given in the local frame that is attached to the starting surface have to be first transformed into a representation in the so-called curvilinear frame. The curvilinear frame is a right-handed coordinate system that follows the trajectory and is mainly characterized by the fact that the plane constructed by the first two coordinate axes is perpendicular to the track direction. Given a track parameter with global position $\vec{m}$ and momentum $\vec{p}$ the axis unit vectors $(\vec{u}, \vec{v}, \vec{t})$ of the curvilinear frame can be constructed as

$$\vec{t} = \frac{\vec{p}}{|\vec{p}|}, \qquad \vec{u} = \frac{\vec{z} \times \vec{t}}{|\vec{z} \times \vec{t}|}, \qquad \vec{v} = \vec{t} \times \vec{u}, \tag{1}$$

where $\vec{z}$ is the vector of the global z axis.

Let L_i denote the 5×5 covariance matrix at the starting surface, and $J_{i,c}$ the Jacobian ensuring the transformation between the coordinates of the local frame and the initial curvilinear frame. The covariance matrix C_i in the initial curvilinear frame can then be expressed by

$$C_i = J_{i,c}^T \cdot L_i \cdot J_{i,c}. \tag{2}$$

The Jacobian $J_{ie,c}(s)$ for the transportation of the covariance matrix between two curvilinear frames is in case of a linear or helical track model only dependent on the track length s and the orientations of the two curvilinear frames. Therefore the covariance matrix C_e in the target curvilinear frame can be written as

$$C_e = J_{ie,c}^T \cdot C_i \cdot J_{ie,c}. \tag{3}$$

Transforming the covariance matrix from the end curvilinear frame to the local frame of the destination surface by using an appropriate transformation then yields the transported covariance matrix L_e in the local representation.

The propagation AlgTools in the ATLAS track extrapolation package are also capable of performing a numerically estimation of the propagated covariance matrix: may $(\lambda_{\mu,i}) = (\lambda_{0,i}, \lambda_{1,i}, \lambda_{2,i}, \lambda_{3,i}, \lambda_{4,i}, \lambda_{5,i})$ be a set of five track parameters on an initial surface and (δ_μ) a set of five chosen numerical values. The Jacobian respecting the transformation between the initial surface and the destination surface can then be calculated by performing six independent propagations to the destination surface. Let $\lambda_{\mu,e}$ denote the set of propagated track parameters and consequently $\lambda_{\mu,e,\nu}$ the set of propagated track parameters when the initial track parameter λ_ν, i has been corrected by the value δ_ν. The coefficients $[J_{ie}]_{(\mu,\nu)}$ of the Jacobian describing the transformation between the initial and the des-

Table 1: Propagation AlgTools

Propagation AlgTool	**Track Model**
StraightLinePropagator	straight line
HelixPropagator	helical
RungeKuttaPropagator	stepwise helical

tination surface then can be expressed as

$$[J_{ie}]_{(\mu,\nu)} = \frac{\lambda_{\mu,e,\nu} - \lambda_{\mu,i}}{\delta_\nu}, \tag{4}$$

and the covariance matrix L_e at the destination surface can straight-forwardly be written as

$$L_e = J_{ie}^T \cdot L_i \cdot J_{ie}. \tag{5}$$

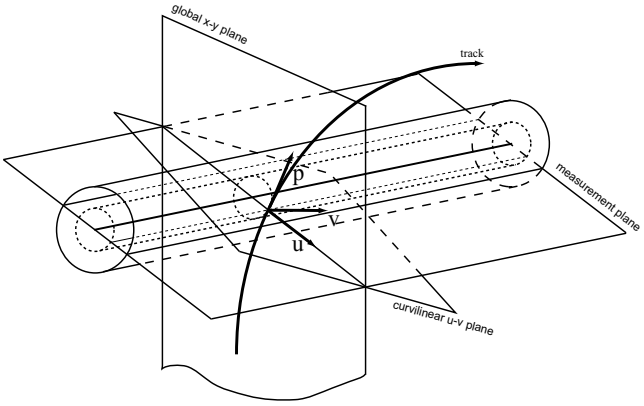

Figure 5: The construction of the curvilinear frame for a measurement on a straw or tube-like surface. This involves the estimation of the measurement plane first in which the error of the measurement is given. The transformation of the covariance matrix given in the measurement plane to the curvilinear frame can then be handled exactly as a transformation of a measurement on a planar surface to the corresponding curvilinear frame.

VISUALIZATION

A visualization of the tracking detector description, the helix class and the common track class has been enabled by embedded converters for the HepVis Event Viewer of the ATLAS detector description package GeoModel. The HepVis Event Viewer is based on the open source 3D modelling toolkit Open Inventor [5]. A screen shot of a visualized propagation process can be seen in Fig. 6.

CONCLUSIONS

The ATLAS track extrapolation package has been extensively tested by reconstructing tracks from simulated and collected data of the ATLAS Combined Testbeam 2004. The tracks have been produced either a global χ^2 fitter or by using a Kalman fitter formalism [6]. During both processes the extrapolation package has been used. Figure 7 shows an example plot of a track residual for one of the installed pixel modules.

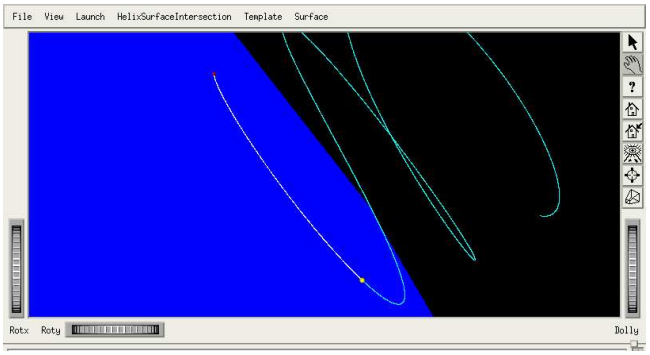

Figure 6: Snapshot of a sample propagation to an arbitrarily oriented plane using the HelixPropagator AlgTool and the HepVis Event viewer of the ATLAS detector description package GeoModel.

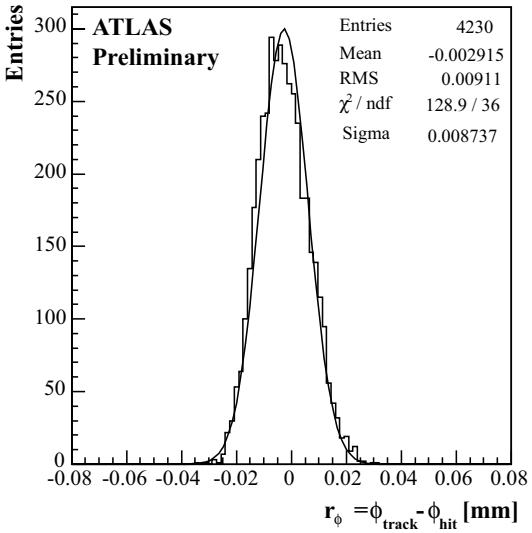

Figure 7: Track residual $r_\phi = \phi_{track} - \phi_{hit}$ in ϕ direction for the fourth pixel module installed in the ATLAS Combined Testbeam 2004.

REFERENCES

[1] http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/index.html.

[2] V. Boisvert *et al.*, "Final Report of the ATLAS RTF", Atlas note ATL-SOFT-2003-010.

[3] Barrand G. *et al.*, "GAUDI - A software architecture and framework for building LHCb data processing applications", Proc. of CHEP 2000.

[4] Calafi ura P. *et al.*, "The StoreGate: a Data Model for the ATLAS Software Architecture", Proc. of CHEP 2003.

[5] http://oss.sgi.com/projects/inventor/

[6] R. Frühwirth, "Application of Kalman fi ltering to track and vertex fi tting", Nucl. Inst. and Meth. A 262 (1987) 444.