GLAS-PPE/2007-24
LHCb Note 2007-140
TRACKING

The LHCb Track Extrapolator Tools

E. Bos

Nikhef, Amsterdam, The Netherlands

E. Rodrigues

University of Glasgow, U.K.

November 2007

Abstract

This note describes the LHCb track propagation models and their implementation in the track extrapolator tools. Their usability and configuration options are discussed. The code referred to is that associated with Brunel release v31r11.

Contents

1 Introduction

The path of a particle traversing the LHCb detector is reconstructed by the software of the Brunel project [1]. This software performs several tasks, including the pattern recognition and the track fit. Pattern recognition in the tracking system deals with assigning the detected hits to tracks. These collections of hits are fitted by a Kalman filter [2] to obtain a representation of the trajectory of the corresponding particle through the detector volume.

As a result of the fit, a track contains the optimal estimates of the track's defining parameters at a set of locations specified by their z-coordinate. In the Kalman filter approach the track parameters at the next z-position of interest are predicted based on the knowledge of the track parameters at the present z-position. This propagation step is performed according to a mathematical model which describes how the track parameters and the corresponding covariance matrix evolve, taking into account the influences of the magnetic field, and the energy loss and multiple scattering due to traversing material. The propagation model is based on the one designed by the Hera-B collaboration [3]. Depending on the specific use-case and the local magnetic field strength, a linear, parabolic, a fifth-order Runge-Kutta or analytic power series expansion track evolution description can be used.

The mathematics of the propagation models is described in section 2. These concepts have been coded up in software tools named track extrapolators. Their technical implementation in the LHCb software is described in section 3.

2 Propagation models

The propagation of the track parameters through the LHCb detector volume is performed in accordance with a suitable selection from the range of available models. Calculating the change of the track parameters in the presence of a sufficiently weak and homogeneous magnetic field is possible with a less complicated and consequently less CPU-intensive model at a comparable accuracy as provided by a model which takes magnetic field effects into account in a more detailed way. Also, for short distances a more accurate model does not provide a significantly different result. The impact of multiple scattering and energy loss on the track parameters and track covariance matrix is modelled separately from the influence of the magnetic field.

These models and the equations of motion which they solve are discussed next.

2.1 Equations of motion

The trajectory of a charged particle through a static magnetic field is described by the equations of motion. In the absence of material effects, these are fully determined by the Lorentz force. Formulated in terms of geometrical quantities and as a function of the z-coordinate, the equations of motion are [4]:

$$\begin{aligned}
\frac{d^2x}{dz^2} &= \frac{Q}{P}\frac{ds}{dz}\left[\ \frac{dx}{dz}\frac{dy}{dz}B_x - (1+\left(\frac{dx}{dz}\right)^2)B_y + \frac{dy}{dz}B_z\right] \quad , \\
\frac{d^2y}{dz^2} &= \frac{Q}{P}\frac{ds}{dz}\left[-\frac{dx}{dz}\frac{dy}{dz}B_y + (1+\left(\frac{dy}{dz}\right)^2)B_x - \frac{dx}{dz}B_z\right] \quad ,
\end{aligned} \tag{1}$$

where Q is the charge of the particle, P is its momentum, s is the path length along its trajectory and $\mathbf{B}$ is the local magnetic field vector.

Rewriting these equations using the LHCb track event model [5] parameters $(x, y, t_x, t_y, \frac{Q}{P})$ results in:

$$\begin{aligned}
\frac{dx}{dz} &= t_x \quad , \\
\frac{dy}{dz} &= t_y \quad , \\
\frac{dt_x}{dz} &= \frac{Q}{P}\sqrt{1+t_x^2+t_y^2}\left[\ t_y(t_xB_x+B_z) - (1+t_x^2)B_y\right] \quad , \\
\frac{dt_y}{dz} &= \frac{Q}{P}\sqrt{1+t_x^2+t_y^2}\left[-t_x(t_yB_y+B_z) + (1+t_y^2)B_x\right] \quad .
\end{aligned} \tag{2}$$

These equations need to be solved for the track parameters x, y, t_x and t_y in order to determine their change during track propagation.

2.2 Linear model

In regions of negligible magnetic field strength, the particle is assumed to traverse the detector in a straight line. Since the slopes t_x and t_y do not change in that case, the solution of the equations of motion for the geometrical track parameters is uncomplicated:

$$\begin{aligned}
x(z_e) &= x_0 + t_{x0}\Delta z \quad , \\
y(z_e) &= y_0 + t_{y0}\Delta z \quad , \\
t_x(z_e) &= t_{x0} \quad , \\
t_y(z_e) &= t_{y0} \quad .
\end{aligned} \tag{3}$$

The subscript 0 refers to the parameter values before propagation and $\Delta z = z_e - z_0$ is the difference between the z-position after and before the propagation step.

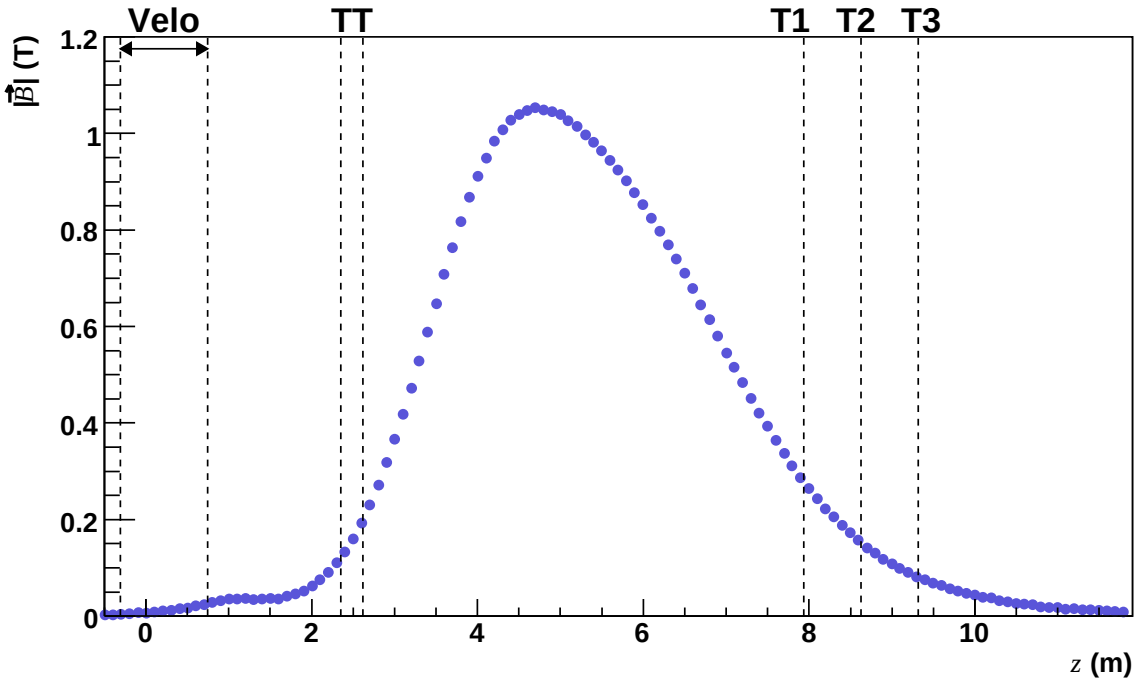

Figure 1: The magnetic field strength as a function of the z-position at $x = y = 0$. The locations of the centres of the tracking detectors are shown as vertical dashed lines. The magnet region is located between the Trigger Tracker (TT) and the T-stations (T1-T3).

2.3 Parabolic model

In the presence of a homogeneous magnetic field, the particle's trajectory can be mathematically described by a parabola. A parabolic model is also an adequate approximation of the particle's path in case of a moderately inhomogeneous magnetic field or when dealing with short propagation distances. This implies that the model can mainly be used for propagation outside of the LHCb magnet, see figure 1.

The equations of motion describing a parabolic trajectory are formulated as follows:

$$\begin{aligned} x(z_e) &= x_0 + t_{x0}\Delta z + \frac{1}{2}\frac{dt_{x0}}{dz_0}(\Delta z)^2 \quad , \\ y(z_e) &= y_0 + t_{y0}\Delta z + \frac{1}{2}\frac{dt_{y0}}{dz_0}(\Delta z)^2 \quad , \\ t_x(z_e) &= t_{x0} + \frac{dt_{x0}}{dz_0}\Delta z \quad , \\ t_y(z_e) &= t_{y0} + \frac{dt_{y0}}{dz_0}\Delta z \quad . \end{aligned} \tag{4}$$

2.4 Runge-Kutta model

A considerable, non-homogeneous magnetic field influences the path of a charged particle in a way which has no exact analytical solution. As seen in figure 1, the LHCb magnetic field inside the magnet region is non-homogeneous and strong enough to disfavour the use of a parabolic propagation model. This situation calls for a numerical solution of the track propagation equations. An iterative method using numerical integration by Runge and Kutta is a well-known technique, suitable for dealing with this problem. The classical implementation is a fourth-order formulation of the approximation to a exact solution. An appropriate solution for the LHCb track propagation purposes is the fifth-order Runge-Kutta method [3]. In this formulation, the track parameter vector $\vec{x} = (x, y, t_x, t_y, Q/P)$ is expanded in terms of Δz [6]:

$$\begin{aligned} \vec{x}(\Delta z) &= \vec{x}_0 + \sum_{m=1}^{6} c_m \vec{k}^m \quad , \\ \vec{k}^m &= \frac{d\vec{x}^m}{dz}\Delta z \quad , \\ \vec{x}^m &= \vec{x}_0 + \sum_{n=1}^{m-1} b_{mn}\vec{k}^n \quad . \end{aligned} \tag{5}$$

The coefficients c_m and b_{mn} are known as the Cash-Karp parameters and are chosen such as to obtain a precision which is proportional to $(\Delta z)^6$, making this a fifth-order method.

2.5 Analytic model

An alternative approach to the Runge-Kutta method has been developed by I. Kisel et al [7]. It is based on the possibility to expand the track parameters which are being propagated in a power series of the magnetic field components. Like the Runge-Kutta formulation, this approach is valid in the presence of an inhomogeneous magnetic field. The inhomogeneity is taken into account by the coefficients of the power series expansion.

The magnetic field components are small parameters in the expansion, resulting in little dependence on the field shape. This feature also allows for higher-order terms of the power series to be neglected without it having a significant impact on the obtained result. In the expansion, same-order terms have vastly different weights. This implies that for a specific order of the expansion, using the few terms with a relatively high weight is sufficient to represent the contribution of that order to the result of the propagation. The desired precision of this model can be controlled by the number of orders taken into account.

The slopes t_x and t_y can be obtained to any order (n) from the following equation, when substituted for T [7]:

$$\begin{aligned} T(z_e) &= T(z_0) + \sum_{k=1}^{n} \sum_{i_1,\ldots,i_k} T_{i_1,\ldots,i_k}(z_0) \\ &\quad \times \left(\int_{z_0}^{z_e} B_{i_1}(z_1) \ldots \int_{z_0}^{z_{k-1}} B_{i_k}(z_k) dz_k \ldots dz_1 \right) \\ &\quad + \mathcal{O}\left(\frac{(B(\frac{Q}{P})\Delta z)^{n+1}}{(n+1)!} \right) \quad , \end{aligned} \tag{6}$$

where the last term in this equation provides an estimate of the propagation error in t_x and t_y, being of order $(n+1)$.

Once the track slopes have been determined, the change in the x and y parameters can be calculated by respectively integrating t_x and t_y over the propagation range in z:

$$\begin{aligned} x(z_e) &= x(z_0) + \int_{z_0}^{z_e} t_x(z) dz + \mathcal{O}\left(\frac{(B(\frac{Q}{P})\Delta z)^{n+1}}{(n+1)!} \right) \Delta z \quad , \\ y(z_e) &= y(z_0) + \int_{z_0}^{z_e} t_y(z) dz + \mathcal{O}\left(\frac{(B(\frac{Q}{P})\Delta z)^{n+1}}{(n+1)!} \right) \Delta z \quad . \end{aligned} \tag{7}$$

2.6 Multiple scattering model

A charged particle traversing the material of the LHCb detector can scatter in the Coulomb field of the nuclei. Given the relative mass difference between the particle and the nucleus, this will be an elastic scattering, changing only the particle's direction, not the absolute value of its momentum.

In case of a thin material layer, the scattering changes the angles without significantly affecting the particle's displacement, due to the limited distance which it travelled. This scenario is modelled by adding contributions to the t_x and t_y variances and their covariance in the track covariance matrix, reflecting the reduced knowledge of the particle's direction. These additions are made after the prediction step:

$$\begin{aligned} cov(t_x, t_x) &= (1 + t_x^2)(1 + t_x^2 + t_y^2)C_{MS} \quad , \\ cov(t_y, t_y) &= (1 + t_y^2)(1 + t_x^2 + t_y^2)C_{MS} \quad , \\ cov(t_x, t_y) &= t_x t_y (1 + t_x^2 + t_y^2)C_{MS} \quad . \end{aligned} \tag{8}$$

The angular distribution corresponding to multiple Coulomb scattering is taken to be the one derived by Molière [8]. Its approximate projected angle distribution is fitted by a Gaussian. The quantity C_{MS} is the square of the root-mean-squared of that fit. The root-mean-squared θ_0 is given by the Highland-Lynch-Dahl formula [9]:

$$\theta_0 = \frac{13.6 \text{ MeV}}{\beta P c} \sqrt{\frac{l\sqrt{1 + t_x^2 + t_y^2}}{\chi_0}} \left[1 + 0.038 \ln \left(\frac{l\sqrt{1 + t_x^2 + t_y^2}}{\chi_0} \right) \right] \quad . \tag{9}$$

The variable $\beta = v/c$ refers to the particle's velocity, P to its momentum, l is the distance in z and χ_0 is the radiation length of the traversed material.

In case of an extended material layer, the effect of the change of angles on the propagated position of the particle can no longer be ignored. For a thick scatterer, the covariance matrix gets additions as defined by the following:

$$\begin{pmatrix} cov(t_x, t_x)\frac{(\Delta z)^2}{3} & cov(t_x, t_y)\frac{(\Delta z)^2}{3} & cov(t_x, t_x)\frac{D\Delta z}{2} & cov(t_x, t_y)\frac{D\Delta z}{2} \\ \dots & cov(t_y, t_y)\frac{(\Delta z)^2}{3} & cov(t_x, t_y)\frac{D\Delta z}{2} & cov(t_y, t_y)\frac{D\Delta z}{2} \\ \dots & \dots & cov(t_x, t_x) & cov(t_x, t_y) \\ \dots & \dots & \dots & cov(t_y, t_y) \end{pmatrix}, \tag{10}$$

where the covariances are given by equations 8 and D is a sign signifying the track direction, being positive for tracks propagating in increasing z.

2.7 Energy loss models

Besides scattering, a particle traversing material also loses energy, mainly by ionisation. This energy loss is accurately described by the Bethe-Bloch equation. When all particles are considered to be minimum ionising particles, the β dependence in the Bethe-Bloch equation can be ignored. In that case the energy loss is given by:

$$\Delta E = -c_{ion}\, \rho\, \frac{Z}{A}\, l \quad , \tag{11}$$

where c_{ion} is the energy loss factor and ρ is the material density.

Landau fluctuations in the energy loss are small compared to the momentum resolution of the detector, eliminating the need to apply corrections to the track covariance matrix. The change in the particle's energy is taken into account by correcting the track's momentum parameter:

$$\begin{aligned} \frac{Q}{P}(z_e) &= \frac{1}{\left(\frac{P}{Q}\right)_0 \pm \delta} & \left(\tfrac{Q}{P}\right)_0 > 0 \rightarrow + \\ & & \left(\tfrac{Q}{P}\right)_0 < 0 \rightarrow - \end{aligned} \tag{12}$$

$$\delta = \pm c_{ion}\,\rho\,\frac{Z}{A}\,\Delta z\,\sqrt{1+t_{x0}^2+t_{y0}^2} \qquad \begin{array}{ll} \text{z decreasing} & \rightarrow + \\ \text{z increasing} & \rightarrow - \end{array} \quad .$$

The energy loss of electrons is treated with a separate model. Rather than ionisation, bremsstrahlung is the dominant cause of energy loss for electrons. The change in energy when passing through a material layer is given by:

$$\Delta E = -E\left(1 - e^{-\frac{l}{\chi_0}}\right) \quad . \tag{13}$$

For electrons, both the momentum and its variance need to be updated at the end of a prediction step through material:

$$\begin{aligned} \frac{Q}{P}(z_e) &= \left(\frac{Q}{P}\right)_0 \times e^{\pm\frac{\Delta z}{\chi_0}\sqrt{1+t_{x0}^2+t_{y0}^2}} & \text{z decreasing} &\rightarrow - \\ & & \text{z increasing} &\rightarrow + \\ cov\left(\frac{Q}{P},\frac{Q}{P}\right) &= \left(\frac{Q}{P}\right)_0^2 \times \left(e^{\pm\frac{\ln 3}{\ln 2}\frac{\Delta z}{\chi_0}\sqrt{1+t_{x0}^2+t_{y0}^2}} - e^{\pm 2\frac{\Delta z}{\chi_0}\sqrt{1+t_{x0}^2+t_{y0}^2}}\right) \quad . \end{aligned} \tag{14}$$

3 Extrapolation tools

As described in section 2, tracks evolve in space following a propagation method. Different models may be used depending on the specific needs of the user and the local magnetic field properties. The propagation models are implemented in a collection of Gaudi tools named track extrapolators, which are located in the `Tr/TrackExtrapolators` package [10].

These track extrapolation tools can propagate a track to a user-specified position and provide the user with the track parameters, the corresponding covariance matrix, and optionally the transport matrix [1] at that position. The track parameters are stored in a vector called a state vector $\vec{x} = (x, y, t_x, t_y, \frac{Q}{P})$, which is stored on the track together with the covariance and transport matrices and the z-coordinate at predefined z-locations in track states.

All track extrapolation tools derive from a common interface and an intermediate base class. The inheritance diagram for the track extrapolators is shown in figure 2 and their implementations are discussed in the following sections.

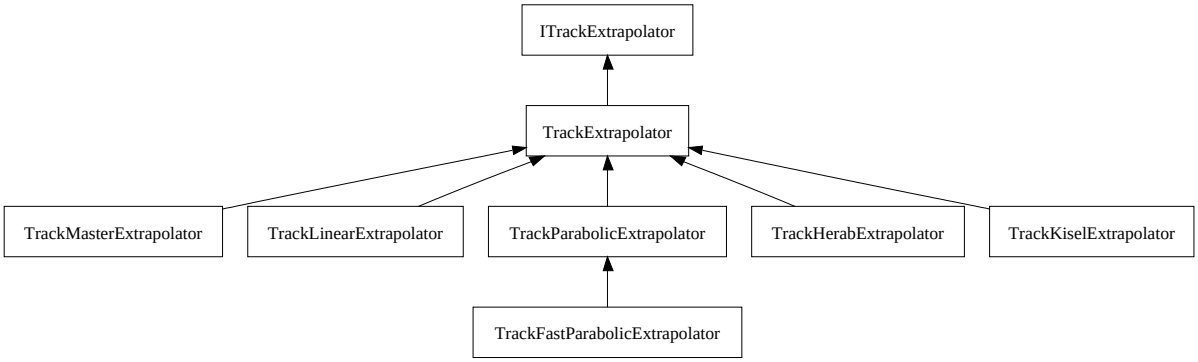

Figure 2: Inheritance diagram for the track extrapolators.

3.1 Extrapolator interface

The `ITrackExtrapolator` tool interface for the extrapolators declares the method signatures common to all concrete extrapolator classes. The code of the interface class can be found in the `Tr/TrackInterfaces` package [11].

The complete set of method signatures, which comprises both the so-called "propagation" and "access" methods, is presented in tables 1 and 2.

[1] The transport matrix is defined as the transformation matrix from the vector of track parameters before to the vector after propagation.

virtual StatusCode propagate (	const Track& track, double z, State& state, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	const Track& track, const XYZPoint& point, State& state, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	const Track& track, Plane3D& plane, State& state, double tolerance = 0.01, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	State& state, double z, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	State& state, double z, TrackMatrix* transMat, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	State& state, const XYZPoint& point, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	State& state, Plane3D& plane, double tolerance = 0.01, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	StateVector& state, double z, TrackMatrix* transportmatrix=0, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	TrackVector& stateVec, double zOld, double zNew, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode propagate (	TrackVector& stateVec, double zOld, double zNew, TrackMatrix* transMat, ParticleID pid = ParticleID(211)) = 0;

Table 1: Signatures of the propagation methods of the extrapolator tool interface `ITrackExtrapolator`. The default particle identity "ParticleID(211)" refers to a pion.

virtual StatusCode positionAndMomentum (	const Track& track, double z, XYZPoint& pos, XYZVector& mom, SymMatrix6x6& cov6D, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode positionAndMomentum (	const Track& track, double z, XYZPoint& pos, XYZVector& mom, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode position (	const Track& track, double z, XYZPoint& pos, SymMatrix3x3& errPos, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode position (	const Track& track, double z, XYZPoint& pos, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode slopes (	const Track& track, double z, XYZVector& slopes, SymMatrix3x3& errSlopes, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode slopes (	const Track& track, double z, XYZVector& slopes, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode p (	const Track& track, double z, double& p, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode pt (	const Track& track, double z, double& pt, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode momentum (	const Track& track, double z, XYZVector& mom, SymMatrix3x3& errMom, ParticleID pid = ParticleID(211)) = 0;
virtual StatusCode momentum (	const Track& track, double z, XYZVector& mom, ParticleID pid = ParticleID(211)) = 0;

Table 2: Signatures of the access methods of the extrapolator tool interface `ITrackExtrapolator`.

The propagation methods listed in table 1 can propagate tracks, track states, state vectors and "statevectors", which are pairs of state vectors and their z-coordinates. In the LHCb tracking software, the corresponding classes are `Track`, `State`, `TrackVector` and `StateVector`, respectively.

Track and state propagation destinations can be specified by a z-coordinate, a point (`XYZPoint`) in space or a plane (`XYZPlane`). The latter includes the option of specifying a tolerance for the acceptable distance between the propagated state vector and the plane. There also exist signatures for the extrapolation of a state vector or a `StateVector` class to a z-coordinate.

Some of the methods take a pointer to a transport matrix (`TrackMatrix`) as well. If a NULL pointer is supplied, then the transport matrix is not calculated, resulting in a faster extrapolation. Otherwise the requested transport matrix is filled with the information from the propagation operation.

The access methods listed in table 2 provide direct information on the parameters of the propagated track. They require the track and z-position of interest as input. In addition one must supply references to objects of the proper type to be filled with the desired information by the access method. These objects are points for positions, vectors for momenta and matrices for covariances. The methods determine which is the closest state on the track to the specified z-position and propagate there in order to obtain the requested information.

3.2 Extrapolator base class

The extrapolator base class `TrackExtrapolator` implements a number of the propagation methods and all of the access methods declared in the interface. Its propagation methods interpret the information given as arguments and use it to formulate a call to the next method in the delegation tree. For instance, all of the methods taking a track as an argument determine the state on that track which is closest to the propagation destination and pass the call on to the appropriate state propagation method. The delegation flow of propagation calls is shown in figure 3.

All of the calls end up at the state vector to z-coordinate with optional transport matrix method or at the state-to-point method. These are not implemented in the base class, but in the deriving classes, which use the specific propagation models described in section 2 for calculating the propagated track parameters.

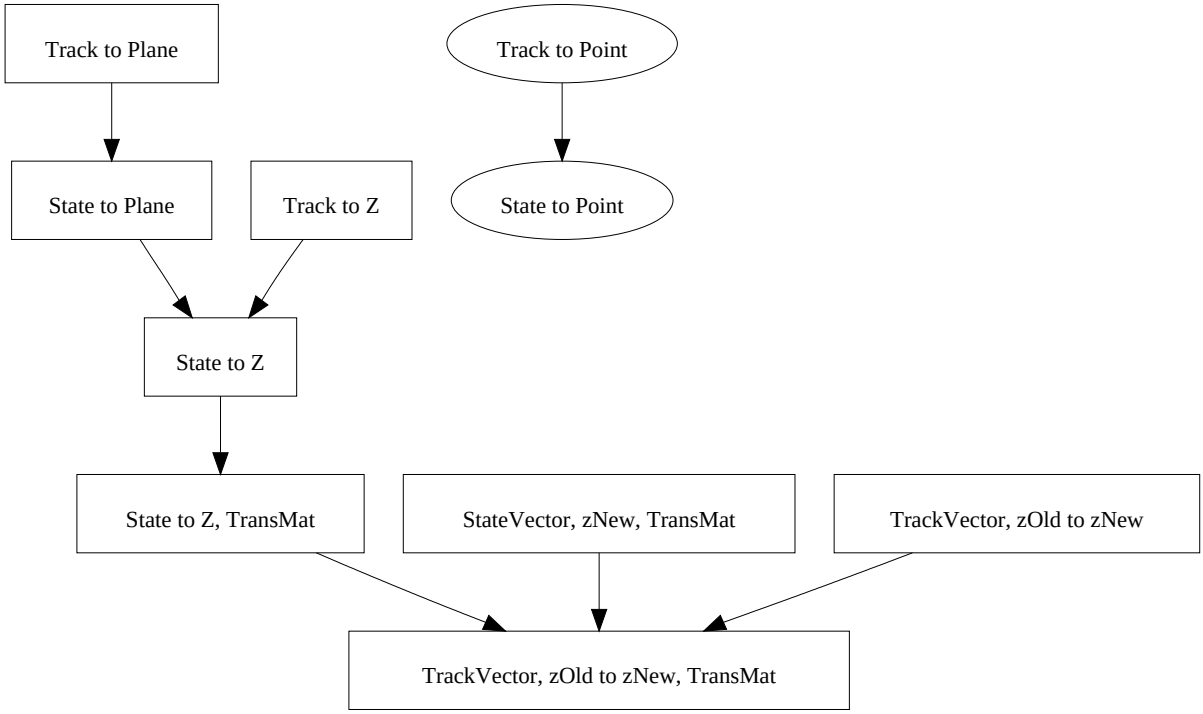

Figure 3: Delegation flow diagram of propagation methods.

A special case is the state-to-plane propagation method. It is implemented as a recursive algorithm, applying the following steps:

1. determine the length of the plane's normal to the present position;
2. stop if that distance is smaller than the user-defined tolerance;
3. determine the origin point of the plane's normal to the present position;
4. propagate the state to the z-coordinate of that point;
5. iterate the above until the distance from the extrapolated state to the plane is less than the tolerance value (user input, 10 μm default). The configurable tool option `Iterations`, see table 3, specifies the maximum allowed number of such iteration steps.

Option name	**Description**	**Default value**
Iterations	Maximum number of iterations when propagating to a plane	5

Table 3: Set of user-definable options of the `TrackExtrapolator` tool base class.

3.3 Master extrapolator

The collection of extrapolator tools contains a special extrapolator, named the "master" extrapolator, `TrackMasterExtrapolator`. It is the only one which takes multiple scattering and energy loss effects into account, as described by the models in subsections 2.6 and 2.7. All of the calculations required for determining which material the particle traverses, what multiple scattering and energy loss corrections are to be applied and how the magnetic field changes the propagated track parameters, are taken care of by separate tools. The choice of tools is steered by job options, which are given in table 4.

The `MaterialLocator` job option of the master extrapolator specifies which tool to use for determining the materials encountered along the track's path during the propagation step. There are two of these tools, both deriving from a common base class called `MaterialLocatorBase`:

1. `DetailedMaterialLocator`:
   locates the materials using the full detector geometry description.

2. `SimplifiedMaterialLocator`:
   locates the materials using a simplified geometry description.

The material locators provide a list of positions of the material layers crossed by the track. The extrapolation is then performed in steps containing at most one material layer. For each step an extrapolator as well as multiple scattering and energy loss correction tools are called.

The master extrapolator uses a tool to decide which specific extrapolation model to use for a propagation step. There is a set of selector tools available for making this decision:

1. `TrackSimpleExtraSelector`:
   automatically selects the parabolic model for extrapolation and refers directly to the `TrackParabolicExtrapolator` tool.

2. `TrackDistanceExtraSelector`:
   selects the parabolic propagation model (`TrackParabolicExtrapolator` tool) for distances shorter than $100\ \mathrm{mm}$, otherwise the Runge-Kutta model (`TrackHerabExtrapolator` tool).

3. `TrackLongExtraSelector`:
   selects the parabolic propagation model in the T-stations region and for distances shorter than $100\ \mathrm{mm}$, otherwise the Runge-Kutta model.

One can choose which selector tool to use by setting the `ExtraSelector` job option, which by default selects the `TrackDistanceExtraSelector`.

Table 4: Set of user-definable options of the main extrapolator tool, `TrackMasterExtrapolator`.

TrackMasterExtrapolator options		
Option name	Description	Default value
MaterialLocator	specify which tool to use for finding material walls	""
ThickWall	treat walls as thick when a t least this thick	0. mm
ExtraSelector	extrapolation model selector tool name	"TrackDistanceExtraSelector"
ApplyEnergyLossCorr	specify whether to appy energy loss corrections	true
ApplyMultScattCorr	specify whether to apply MS corrections	true
ThinMSCorrectionTool	MS correction tool name for thin walls	"StateThinMSCorrectionTool"
ThickMSCorrectionTool	MS correction tool name for thick walls	"StateThickMSCorrectionTool"
GeneralDedxCorrectionTool	general energy loss correction tool name	"StateSimpleBetheBlochEnergyCorrectionTool"
ElectronDedxCorrectionTool	electron energy loss correction tool name	"StateElectronEnergyCorrectionTool"
ApplyElectronEnergyLossCorr	specify whether to apply electron energy loss corrections	true
StartElectronCorr	only apply electron energy loss corrections after this z-coordinate	2500. mm
StopElectronCorr	only apply electron energy loss corrections before this z-coordinate	9000. mm
MaxStepSize	maximum propagation step size in z	1000. mm
MaxSlope	Acceptance in track slope (maximum state t_x- and t_y-slopes)	5.
MaxTransverse	LHCb transverse acceptance for extrapolation (maximum state x- and y-positions)	10. m

Option name	Description	Default value
MSFudgeFactor2	Scaling of θ_0 distribution width	1.0

Table 5: Set of user-definable options of the `StateThinMSCorrectionTool` and `StateThickMSCorrectionTool` tools.

Multiple scattering is taken into account by increasing some of the track's covariance matrix elements, reflecting the reduced precision with which the corresponding propagated track state parameters are known. The master extrapolator's `ApplyMultScattCorr` job option can be set to `true` or `false`, depending on whether the multiple scattering correction is to be applied or not.

As discussed in subsection 2.6, there are differences in how thin and thick material layers are treated. Both approaches have been implemented in tools, which are located in the `Tr/TrackTools` package [12]. They are called `StateThinMSCorrectionTool` and `StateThickMSCorrectionTool`, respectively. Their only job option, shown in table 5, is called `MSFudgeFactor`, which is a tuning variable that scales the width of the θ_0 distribution as defined by equation 9. It mainly impacts the pull distributions of the fitted track parameters and is set to 1 by default.

Through the master extrapolator's `ThickWall` job option, one can select the maximum width of a thin layer, which is set to $0\ \mathrm{mm}$ by default. This implies that equations 10 are used for all material layers during standard operation of the track fit.

The energy loss correction changes the value of the track state's momentum at the end of a propagation step. In case the particle is an electron, the momentum variance is also increased. Whether or not to apply the general energy loss corrections is steered by the `ApplyEnergyLossCorr` option. The `ApplyElectronEnergyLossCorr` option determines whether the specialised electron energy loss corrections are applied to the propagated track.

The corresponding tools are named `StateSimpleBetheBlochEnergyCorrectionTool` and `StateElectronEnergyCorrectionTool`, which are located in the `Tr/TrackTools` package [12]. Tables 6 and 7 shows their job options. The `EnergyLossFactor` option refers to the c_{ion} constant of equation 12. In order to prevent excessive loss of energy, the correction is limited to a maximum value as specified by the `MaximumEnergyLoss` job option, which is set to $100\ \mathrm{MeV}$ by default. The `MaximumRadLength` variable sets a limit to the correction term for the electron energy correction.

The electron energy loss correction is applied to propagation steps which fall within a z-range defined by the `StartElectronCorr` and `StopElectronCorr` job options of the `TrackMasterExtrapolator` tool. This range spans the magnet region, where the magnetic field is strong enough to make the momentum change detectable.

Option name	Description	Default value
EnergyLossFactor	The energy loss multiplication factor	354.1 MeV*mm^2/mole
MaximumEnergyLoss	Maximum allowed energy loss per step	100. MeV

Table 6: Set of user-definable options of the `StateSimpleBetheBlochEnergyCorrectionTool` tool.

Option name	Description	Default value
MaximumRadLength	Maximum value for the radiation length correction factor	10.

Table 7: Set of user-definable options of the `StateElectronEnergyCorrectionTool` tool.

3.4 Linear extrapolator

The linear extrapolator tool, `TrackLinearExtrapolator`, implements the extrapolation method for a state vector to a z-coordinate, with optional determination of the transport matrix. Since the model in this case is a straight line, only the x and y components of the state vector are modified in accordance with equations 3. Analogously, only the x and y slope elements of the transport matrix are filled with the z-displacement value.

The other implemented method is that for the propagation of a state to a point. In that case the displacement in z is determined by calculating the zero-point of the derivative of the distance between the track and the point with respect to the z-coordinate. Subsequently the propagation call is diverted to the first propagation method.

3.5 Parabolic extrapolator

The parabolic extrapolator, `TrackParabolicExtrapolator`, uses equations 4 for track propagation. Inside the state vector propagation method, it determines the magnetic field value at the midway point in z, using the service set by the magnetic field service job option, see table 8. This value is also used in the method which is called for updating the transport matrix. Updating the transport matrix is performed by a separate method such as to simplify the fast parabolic extrapolator, described in subsection 3.6.

Propagating a state to a point when dealing with a parabolic trajectory is performed using a similar approach as for a straight line track. The distance between the track and the point is minimised by determining the z-coordinate at which the derivative of the distance with respect to z is zero and subsequently propagating to this z-coordinate. This however is a cubic equation, having either one or three solutions. In case of three solutions, the solution yielding a z-coordinate closest to the present z-position is selected.

Option name	Description	Default value
FieldSvc	The choice of magnetic field provider	"MagneticFieldSvc"

Table 8: Set of user-definable options of the parabolic extrapolator `TrackParabolicExtrapolator`.

Option name	Description	Default value
requiredPrecision	Desired accuracy of the extrapolation	0.005 mm
FieldSvc	Choice of magnetic field provider	"MagneticFieldSvc"

Table 9: Set of user-definable options of the Runge-Kutta extrapolator `TrackHerabExtrapolator`.

3.6 Fast parabolic extrapolator

The fast parabolic extrapolator, `TrackFastParabolicExtrapolator`, implements the parabolic propagation model, just as the "standard" parabolic extrapolator, but relies on a fast and less accurate determination of the track state parameters covariance matrix. It inherits its propagation methods from the parabolic extrapolator. Only the method which updates the transport matrix differs.

3.7 HERA-B extrapolator

The "HERA-B" extrapolator, `TrackHerabExtrapolator`, implements a fifth-order Runge-Kutta method as described in subsection 2.4. It uses the state vector expansion of equations 5 to numerically calculate the change in the geometrical track parameters between two z-coordinates.

The size of the steps in z determines the precision with which the change in magnetic field strength is taken into account for the propagation. A fourth-order Runge-Kutta result is calculated after the fifth-order one by using an alternative set of c_m coefficients in equations 5. The difference in their x and y determinations must not exceed a maximum value, which is specified by the job option `requiredPrecision`, see table 9. When the step size is determined to be too large, it is halved and the track parameters are recalculated.

3.8 Kisel's analytic extrapolator

The analytic formula 6 for the slopes has been implemented to third order in the tool `TrackKiselExtrapolator`, which provides equivalent performance to a fourth order Runge-Kutta solution. It samples the magnetic field at the beginning, middle and end of the propagation step, integrating to get the values in between. Like the

Option name	Description	Default value
order	Expand the slopes to this order	3
FieldSvc	Choice of magnetic field provider	"MagneticFieldSvc"

Table 10: Job options of the analytic extrapolator `TrackKiselExtrapolator`.

`TrackHerabExtrapolator` extrapolator, only the state vector to a z-position propagation method has been implemented. A fourth order expansion may be coded at a later point in time. In that case, this extrapolator is expected to be equivalent to the Runge-Kutta extrapolator. The order of the expansion can then be set through the `order` job option, see table 10.

References

[1] The LHCb event reconstruction application project page
http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/brunel/

[2] E. Rodrigues, *The LHCb Track Kalman Fit*, LHCb Note 2007-014, 2007

[3] R. Mankel, *Application of the Kalman Filter Technique in the Hera-B Track Reconstruction*, HERA-B note 95-239, 1995;
R. Mankel, *Ranger - A pattern recognition algorithm for the HERA-B main tracking system. Part IV: The object-oriented track fit*, HERA-B note 98-079, 1998

[4] R. Frühwirth et al, *Data Analysis Techniques for High-Energy Physics*,
Cambridge University Press, 2000

[5] J. A. Hernando and E. Rodrigues, *Tracking Event Model*,
LHCb Note 2007-007, 2007

[6] W. H. Press et al, *Numerical recipes : the art of scientific computing*,
Cambridge University Press, 2007

[7] S. Gorbunov, I. Kisel,
Analytic formula for track extrapolation in non-homogeneous magnetic field,
Nucl. Instr. and Meth. A559, 148-152, 2006

[8] G. Molière, *Z. Naturforsch*, 2a, 133, 1947; 3a, 78, 1948

[9] G. R. Lynch and O. I. Dahl, *Approximations to Multiple Coulomb scattering*,
Nucl. Instr. and Meth. B58, 1991.

[10] Track extrapolators package at CVS repository
http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/Tr/TrackExtrapolators/?cvsroot=lhcb

[11] Track interfaces in package at CVS repository
http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/Tr/TrackInterfaces/?cvsroot=lhcb

[12] Tracking tools in package at CVS repository
http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/Tr/TrackTools/?cvsroot=lhcb