CBX 96-20
Robert Kutschke
Anders Ryd
September 2, 1996

Billoir fitter for CLEO II

Abstract

The implementation of a Billoir fitter as a final track fitter for CLEO II is described. This note serves two purposes. First it will describe the idea and the formalism behind the Billoir fit. The second purpose is to provide a detailed documentation of the implementation. The fitter is designed to correctly handle the effects of gaussian multiple scattering and energy loss with gaussian straggling. Correct means that the estimator optimal, unbiassed and that the error matrix is correct. The fitter also has an extension to be able to handle non-uniform magnetic fields. This is not used in CLEO but it has been tested and shown to work satisfactorily.

Contents

1 Introduction

The purpose of this note is to describe how the Billoir fitter [1], intended to be the final track fitter for CLEO II, works. This note will serve both as a users guide and as a technical description of the algorithms and methods used. It will also contain a fairly comprehensive discussion of the concepts and ideas behind the Billoir fitter and general χ^2 fitting.

The goal of this project was to write a final fitter that would 'correctly' treat the effects of multiple scattering and energy loss. This also means that it would have to do separate fits for the five different particle hypothesis: e, μ, π, K, and p, as their interactions with matter are different. Further, there is a time constraint for a fit. The current final fitter, `TF3FIT`, takes approximately 15 ms per fit, on an Alpha 600. A goal was set to not take significantly longer time with the new fitter. However, the new fitter has to perform five fits per track, one fit for each particle hypothesis, and account for the material so the code has to be efficient in order to meet this goal. Some effort has been put on code optimization and some simple things, like to avoid looping over indices, save a significant amount of time.

Another goal was that the fitter should use the material description in `CLEOG`, the `GEANT` based detector simulation package.

Figure 1 illustrates the different building blocks that make up the Billoir fitter. At the heart of the method is the operation of adding a hit to the track. This operation needs to know about the hits and the track parameters. The second operation that is essential for the fitter is the transport of the track parameters. The operation of transport takes track parameters at a given point and expresses them at another point. This requires knowledge about the B-field, the material, particle hypothesis, and so on.

2 The Billoir fit

This section describes how the Billoir fit, also called a Kalman filter, works. The idea as applied to a track fit is quit simple and very beautiful. This section is divided into three subsections. The first section describes very simply the idea of a the Billoir fit. The second section describes in more detail how it works, giving a more mathematical treatment of the problem. Both of these sections do not discuss the important problem of starting the fit, which is the topic of the last section. In this implementation, we have taken the approach of linearizing the fit around a seed track, to obtain stability at the start of the fit. Exactly what this means is described in the last section.

Before plunging in to the world of Billoir fitting some motivation for why we would like to use it might be useful. This discussion will be in the form of a comparison with a standard χ^2-fit for the extraction of track parameters from a set

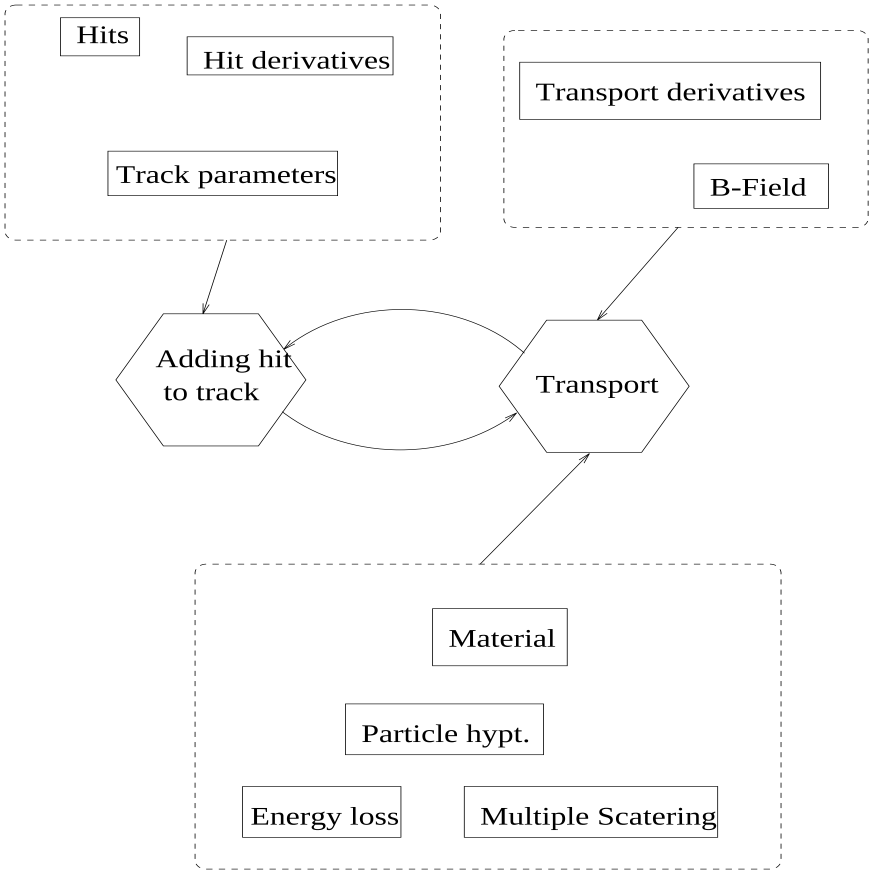

Figure 1: The task of estimating the optimal track parameters and simultaneously correctly estimate the variance matrix for the estimated track parameters involves a solid understanding of both the detector: resolutions, alignment, magnetic field, etc. and the interactions the produced particles have with the material in the detector: multiple scattering and energy loss.

of position measurements of a charged track.

When a charged particle goes in a uniform magnetic field it traces out a helix in space. (A uniform field is considered for simplicity but it is not necessary for the discussion.) The particles trajectory is a function of some set of initial parameters, e.g., its momentum and position when produced. The task of track fitting, in general, is to determine these initial parameters from a set of position measurements along the track. When a particle does not interact with material, an optimal estimate of the initial parameters is given by a simple χ^2-fit to the measured positions. It is assumed that measurements are normally distributed, so that the estimator is optimal (and unbiased). However, if there is material along the track, the (charged) particle will interact, for example by multiple scattering. The simple χ^2-fit must be modified in order to handle this. A more general version of the χ^2-fit which can handle this, requires the inversion of a (symmetric) $N \times N$ matrix, where N is the number of measurements along the track. This becomes a time consuming operation as the computation time needed to invert an $N \times N$ matrix is, for practical purposes, $\mathcal{O}(N^3)$. This is the main disadvantage with the general χ^2 method. The other disadvantage is that it is difficult to calculate the required derivatives of the measurements with respect to the track parameters. This is especially true when the trajectory has a complicated form; for example when there is energy loss along the track or if the magnetic field is not uniform. The Billoir fitter described in this CBX is intended for use in a uniform solenoidal field, but is modified to handle smaller non-uniformities in the magnetic field. As will be shown below the Billoir fitter provides an algorithm for doing the fits with a time complexity of $\mathcal{O}(n)$.

2.1 The idea behind the Kalman filter

We want to estimate a set of parameters, η, from a set of measurements, say of positions along the track. The idea is to add on the information from each of the measurements successively to obtain the best estimate of the track parameters. This necessitates that the procedure is started, which turns out to be slightly non-trivial and will not be discussed untill later. For the present discussion, it is assumed that the fit has been started and we have a best estimate of the parameters, η, and a variance matrix, V. The procedure is to use the next measurement to obtain a new best guess, η^{opt}, of the parameters and an updated variance matrix which contains information from the added measurement. When applied to track fitting, the track parameters must be transported to the location of the next measurement, before it can be added on. Trajectories through material, where the particle may, for instance, multiple scatter, are simply accounted for by updating the variance matrix with the appropriate uncertainties, such that when a new measurement is added the old estimate of the track parameters is de-weighted by the increase of the variance matrix due to the interactions with the material.

The Billoir fit is a χ^2 fit which accounts for the effects of multiple scattering and other sources of noise. This is accomplished without having to invert an $N \times N$ matrix, instead the Billoir fit adds on the information from the measurements one at the time and extracts the current best guess together with a variance matrix. The estimate of the parameters, together with the variance matrix is then transported to the next measurement. If the next measurement is, for instance a hit on a silicon wafer in a vertex, detector the parameters are transported to a point close to the measurement, and the variance matrix is updated with the effects of the material which the track passed through.

The fitting procedure consists conceptually of two parts, as depicted in Figure 2. The first part is to add on the information from a measurement; simply described in the language of the weight matrix (also called the information matrix) which is the inverse of the variance matrix. It is convenient to think in terms of weight matrices since they are always defined, the variance matrix is singular if the system is under-constrained, i.e., if there are more free parameters then measurements.

The second part of the procedure is the concept of transport, in which the parameters and the variance matrix are transported between successive points. This must account for all physics processes that the particle is subjected to. This is most easily described in terms of the variance matrix, as will be evident below. This implies that the process of transporting and adding on new measurements requires an inversion of the matrix in each step. This turns out not to be necessary as the inversions needed can be done analytically, using the so called Woodbury's matrix inversion theorem, described in the next section.

The steps of adding on the information from measurements and transporting the track parameters are described in detail in sections 2.2 and 4 respectively.

2.2 Precise formulation of the Kalman filter

This section will provide a more technical presentation of the ideas described in the previous section.

Assuming that a set of track parameters, η, and their variance matrix, V_η, have been estimated, for example from a earlier set of measurements, as shown in Fig. 2. As described in the section above, the idea of the Billoir fit is to add on one hit at

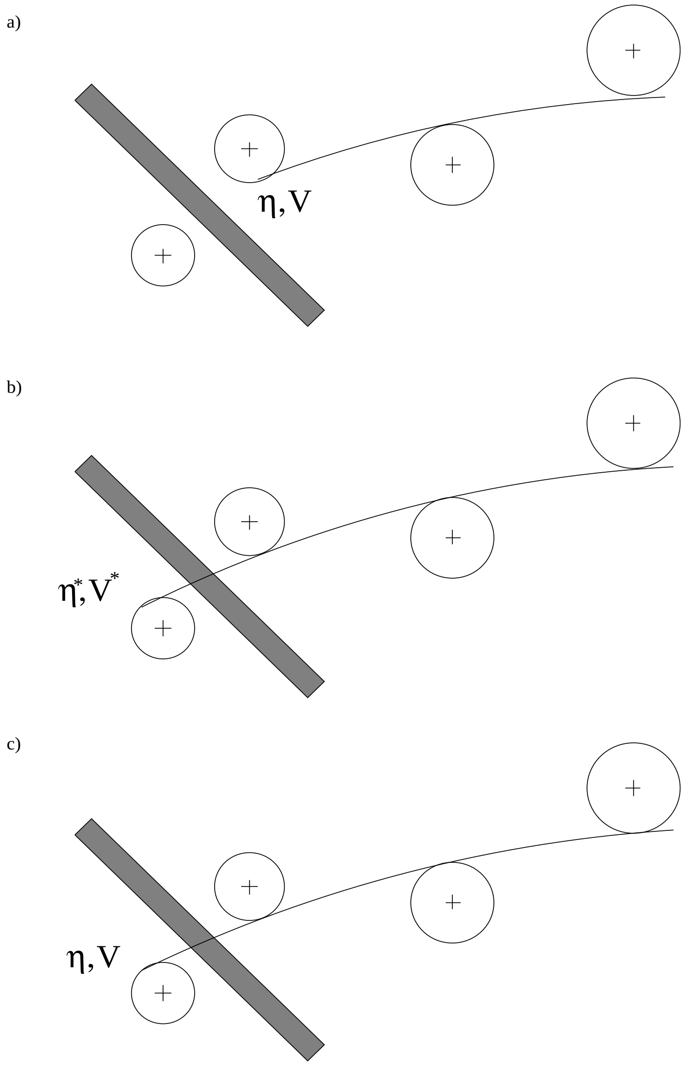

Figure 2: The idea of the Billoir fit. Part a) shows the track with the estimated parameters, η, and the variance matrix, V. This is after the measurements 1,2, and 3 has been added. b) shows the track after it has been transported to measurement 4. The track parameters, η^*, and the variance matrix, V^*, have been transported and updated with the effects of passing through material, shown as the shaded area. In c) the information from the last measurement, 4, has been added to the track and a new best estimate, η, is obtained together with an estimate of the variance matrix.

the time. This leads us to study how the parameters and their variance matrix are affected by the information provided by adding a new measurement.

After a hit is added to the fit we have a new set of track parameters, η^{opt}, with a variance matrix, V_η^{opt}. The hit provides a measurement of some quantity d^{meas}, e.g., a drift distance, with an estimated variance of σ^2. We write, $d = d(\eta^{\mathrm{opt}})$, the predicted value of the measured quantity as a function of the estimated track parameters. We can now form a combined χ^2 which combines the old estimate and the new measurement as

$$\chi^2 = \frac{\left(d(\eta^{\mathrm{opt}}) - d^{\mathrm{meas}}\right)^2}{\sigma^2} + \left(\eta^{\mathrm{opt}} - \eta\right)^t V_\eta \left(\eta^{\mathrm{opt}} - \eta\right). \tag{1}$$

Note that the added measurement is uncorrelated with η, which is estimated from the previously added measurements. To proceed it is convenient to write

$$\eta^{\mathrm{opt}} = \eta + \tilde{\eta} \tag{2}$$

and to linearize $d(\eta^{\mathrm{opt}})$

$$d(\eta^{\mathrm{opt}}) = d(\eta + \tilde{\eta}) = d(\eta) + \frac{\partial d}{\partial \eta}\tilde{\eta} = d(\eta) + D^t \tilde{\eta} \tag{3}$$

where

$$D = \frac{\partial d}{\partial \eta^t} \tag{4}$$

such that

$$\chi^2 = \frac{\left(d(\eta) - d^{\mathrm{meas}} + D^t \tilde{\eta}\right)^2}{\sigma^2} + \tilde{\eta}^t V_\eta \tilde{\eta}. \tag{5}$$

In section 5 the calculation of the derivatives, D, are described for the different types of hits. Minimizing this χ^2 and solving for $\tilde{\eta}$ gives

$$\tilde{\eta} = \left(\frac{d^{\mathrm{meas}} - d(\eta)}{\sigma^2}\right)\left(\frac{DD^t}{\sigma^2} + V_\eta^{-1}\right)^{-1} D. \tag{6}$$

It is now very convenient to apply the so called Woodbury's matrix inversion formula which applied to our case gives

$$\left(\frac{DD^t}{\sigma^2} + V_\eta^{-1}\right)^{-1} = V_\eta - \frac{V_\eta D D^t V_\eta}{\sigma^2 + D^t V_\eta D}. \tag{7}$$

To show that this is correct is a simple exercise of multiplying together the two matrices. It is however a little more complicated to come up with the expression. This is a special case of a little more general formula [1]. Care has to be taken with numerical stability of this equation [2]. Using this we can now write

$$\eta^{\mathrm{opt}} = \eta + \left(\frac{d^{\mathrm{meas}} - d(\eta)}{\sigma^2}\right)\left(V_\eta - \frac{V_\eta D D^t V_\eta}{\sigma^2 + D^t V_\eta D}\right) D. \tag{9}$$

As always we are interested in finding the variance matrix, $V_\eta^{\mathrm{opt}} = \langle \delta\eta^{\mathrm{opt}} \delta\eta^{\mathrm{opt}^t} \rangle$, of the estimated parameters. This is a straight forward, but tedious, calculation. Note that $\delta d(\eta) = D^t \delta\eta$. The result is, as one could guess,

$$V_\eta^{\mathrm{opt}} = V_\eta - \frac{V_\eta D D^t V_\eta}{\sigma^2 + D^t V_\eta D}. \tag{10}$$

If we write this in terms of the weight matrix instead we have

$$W_\eta^{\mathrm{opt}} = W_\eta + \frac{D D^t}{\sigma^2}. \tag{11}$$

This is telling us that the information provided by the measurement is DD^t/σ^2 this is a very natural and particularly easy to understand in the case when one of the measurements is a track parameter.

[1] The general form of Woodbury's matrix inversion formula is $(A + UV^t)^{-1} = A^{-1} - A^{-1}U(1 + V^t A^{-1} U)^{-1} V^t A^{-1}$ where A is an $n \times n$ matrix and U and V are $n \times p$ matrices (with $p < n$). In our application $p = 1$. $(1 + V^t A^{-1} U)$ is an $p \times p$ matrix. The formula is easily proved by multiplying together $(A + UV^t)$ and $A^{-1} - A^{-1}U(1 + V^t A^{-1} U)^{-1} V^t A^{-1}$.

[2] To see when care has to be taken consider the case when the derivative D is zero for all parameters except the i:th. This happens when one of the track parameters is the measured quantity, e.g., when when the track is referenced at the center of an axial wire the drift distance is the inpact parameter. Then it is easy to see that

$$V^*_{\eta_{ii}} = V_{\eta_{ii}} - \frac{V_{\eta_{ii}} V_{\eta_{ii}}}{\sigma^2 + V_{\eta_{ii}}} = \left(\frac{1}{\sigma^2} + \frac{1}{V_{\eta_{ii}}}\right)^{-1} \tag{8}$$

The second equality shows that the new error on the i:th parameters is exactly what expected from averaging. However, if $\sigma^2 \ll V_{\eta_{ii}}$ then numerical precision can cause the evaluation of $V_{\eta_{ii}}$ to give 0. This is a problem when a measurement with good precision (small σ^2) is to be added on to a track that is not well measured (large $V_{\eta_{ii}}$), e.g., at the first addition of a z-measurement in the SVX. This is solved in the code by the use of double precision in this step. As far as we know right now this is the most critical point in the calculation.

This successive addition of the measurements also allows for the construction of a χ^2. The total χ^2, is calculated by updating the χ^2 by the contribution from the addition of the last hit

$$\chi^2_{\mathrm{new}} = \chi^2_{\mathrm{old}} + \frac{(d(\eta) - d^{\mathrm{meas}} + D^t\tilde{\eta})^2}{\sigma^2} + \tilde{\eta}^t V_\eta \tilde{\eta}. \tag{12}$$

The procedure described here gives you the optimal track parameters at the origin or the production point of the track. To obtain the optimal estimate of the track parameters at other points along the track, the fit must be done in two passes, an inward pass and an outward pass and the results averaged along the track. This procedure is called smoothing and is discussed in Section 7.

2.3 Starting the fit:Linearization

So far we have avoided the task of how to start the fit. Intuitively, you want to start with an initial variance matrix that has large errors on the initial track parameters. The errors have to be so large that the seed does not bias the final fit. In principal the errors could be made very large, though not infinite since that would posses a numerical problem.

One catch with this approach, is that the fit might go 'astray', meaning that initial measurements pull the fit in a wrong direction. Instead of converging to the true parameters almost anything might happen. There are two sources for this problem. The first is that the χ^2-estimator is an optimal estimator only for a linear problem, i.e., higher moments are neglected. Secondly the transport will become more or less undefined when the track parameters to transport are not describing the real track. A typical scenario is when the first few (3) measurements estimate a very large curvature, i.e., low momentum. Then the transport to, and adding on, the next hit is not well defined, as shown in Figure 3. It has been suggested, to start the fit with a variance matrix having some weight on the initial seed so that the first few measurements will not pull the fit astray. This has to be done with a great deal of care, as the initial weight can not be so big that it affects the final fit, i.e., the seed should not bias the final result. This method is not used in this fitter, instead, we have taken another approach mentioned by Fruhwirth [6].

The approach we have taken is to linearize the fit around the seed track. This has two advantages. It linearizes the fit so that the χ^2 estimator becomes 'correct' and it allows the transport to be made along the seed track, so that it will not be sensitive to the initial measurements if they pull the fit away from the true value.

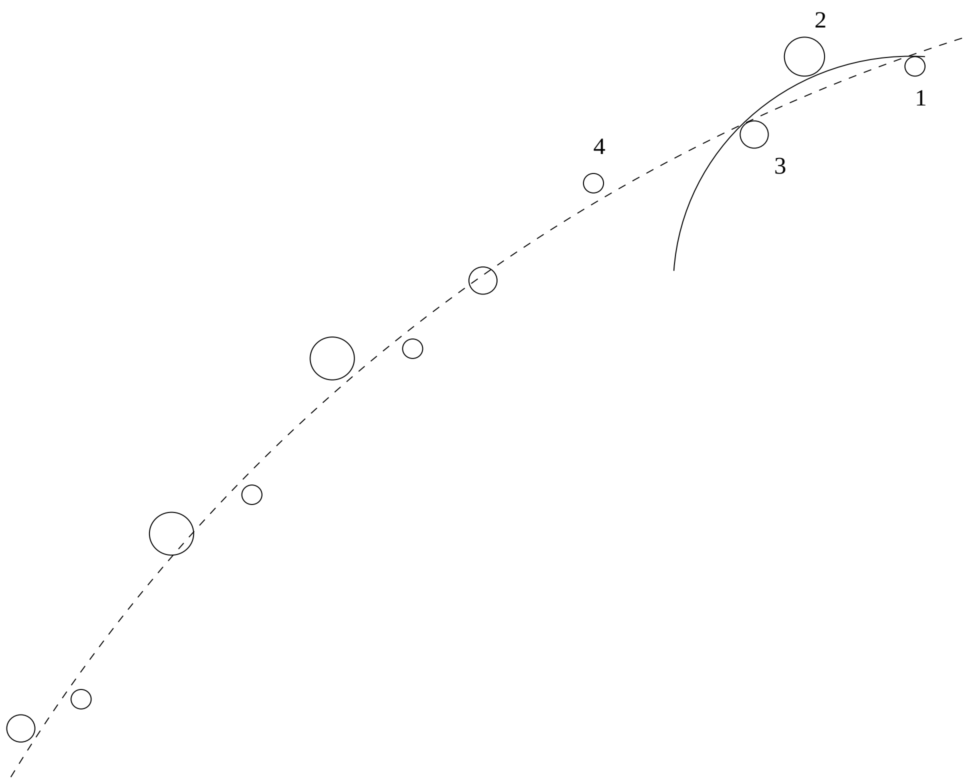

Figure 3: This figure illustrates a problem that can occur at the start of a fit. After the three first hits are added the estimate of the track parameters is not good. This might be caused by a hit that does not belong on the track or simply a hit with a large residual. After the first three hits are added the estimate of the track is indicated with the solid line. The actual track is indicated by the dashed line. When the track parameters are transported from the third hit to the fourth the transport is almost meaningless since the estimate of the track is far from what the actual track is. This is solved by linearizing the track around the seed.

In this linearization the track is described as a perturbation, $\Delta\eta$, around a seed track, η_0,

$$\eta = \eta_0 + \Delta\eta. \tag{13}$$

It is important how we define the transport of this track and how information is added on to the track using the linearized track model.

Let us first discuss the transport since it is very straight forward. The seed track, η_0, is transported in the standard way to obtain η_0'. And as usual we define the transport matrix, T, as

$$T = \frac{\partial\eta_0'}{\partial\eta_0}. \tag{14}$$

Then $\Delta\eta$ is transported linearly as

$$\Delta\eta' = T\Delta\eta. \tag{15}$$

When a hit is added to the fit two things are needed: the predicted measurement, $d(\eta)$, from the track parameters as well as the derivatives of this measurement with respect to the track parameters. The predicted measurement, $d_0 = d(\eta_0)$, from the seed track is calculated and the derivatives, D, of this distance with respect to the track parameters are calculated. The predicted measurement of the track is then evaluated as

$$d = d_0 + D^t\Delta\eta. \tag{16}$$

When the track is traced through the material, the path and the current momentum of the particle are taken from the seed track. This very simple procedure has shown to be stable against the problems discussed above. The initial error matrix can be arbitrarily large.

3 Track model

This section describes the conventions that are used in the description of the track in the Billoir fitter. The track model is, locally, a helix, described by five track parameters, $\eta = (\mathrm{K}, \phi, d, t, z)$. This discussion follows very closely that of Ref. [3], but it is included here to make this document self contained.

K is the signed curvature ($= q/2\rho$), where ρ is the radius of curvature and q is the charge in units of the proton charge. Note that K is the "diameter of curvature". The sign of the curvature is the same as the charge of the particle. K is measured in m^{-1}.

ϕ is the azimuth of the momentum, P, at the point of closest approach of the track to the z-axis.

d is the signed 2-D distance of closest approach of the track to the z-axis. The sign convention is as follows. Complete the circle of the track in the $r-\phi$ $(x-y)$ plane. If the origin is outside the circle then d and K have the same sign.

t is the tangent of the dip angle, $\tan\lambda$. Also often expressed as the cotangent of the polar angle, $\cot\theta$.

z is the value of z at the point of closet approach of the track to the z-axis.

In the `DUET` based tracking, the track parameters are only refered to in one coordinate system, the global CLEO coordinate system with the origin in the center of the detector. In the Billoir fitter, it is very convenient to describe the track with respect to different coordinate systems along the track. For example it is very convenient to transport to the wire position when adding on an axial wire hit to the track. However, these other coordinate systems are only linear translations, not rotations, of the global coordinate system. This means that the coordinate axes in the local coordinate system are parallel to the axes of the global coordinate system.

From the track parameters and the magnetic field strength we can calculate the components of the particle's 3-momentum:

$$P_x = \frac{A}{|\mathrm{K}|}\cos\phi, \tag{17}$$

$$P_y = \frac{A}{|\mathrm{K}|}\sin\phi, \tag{18}$$

$$P_z = \frac{A}{|\mathrm{K}|}t, \tag{19}$$

where $A = cB/2$. c is the speed of the light and B is the magnetic field, the factor of 2 comes from the fact that K is the "diameter of curvature". With a 1.5 Tesla magnetic field we get $A = 0.2248425$ GeV/mc. This is in units such that the momentum is expressed in GeV/c and K is in m^{-1}.

For many purposes it is useful to parameterize the helix in terms of the track parameters. This will be used repeatedly throughout this work for calculation of derivatives, intercepts of tracks with detector elements, etc. The helix is most easily parameterized in terms of the turning angle, ψ, from the point of closest approach of the track to the z-axis:

$$
\begin{aligned}
x_t(\psi) &= -d\sin\phi + \frac{1}{2\mathrm{K}}(\sin(\phi+q\psi) - \sin\phi), && (20)\\
y_t(\psi) &= \quad d\cos\phi + \frac{1}{2\mathrm{K}}(\cos\phi - \cos(\phi+q\psi)), && (21)\\
z_t(\psi) &= z + t\rho\psi = z + \frac{q\psi}{2\mathrm{K}}t. && (22)
\end{aligned}
$$

This is generally a very convenient form, however, as the curvature, K, goes to zero, ψ goes to zero for a constant arc length along the track. The equations are well behaved in this limit but they become numerically instable. This problem can be solved by parameterizing the track as a function of the arc length instead of the turning angle. Using this the track parameterization becomes numerically stable even as the curvature goes to zero, i.e., for a straight line. This is a nice feature, and is especially desirable in patter recognition. However, it is not a problem in the current implementation of the fitter, as all operations, like transport and evaluation of derivatives are done for a seed track which is well behaved.

It is sometimes convenient, e.g., for plotting, to have the center of the track circle in the xy-plane:

$$
\begin{aligned}
x_c &= -(d + \frac{1}{2\mathrm{K}})\sin\phi, && (23)\\
y_c &= \quad (d + \frac{1}{2\mathrm{K}})\cos\phi. && (24)
\end{aligned}
$$

4 Track parameter transport

This section will discuss how track parameters and their variance matrix are transported. The transport is divided into two steps, the first being simply a change of variables. If a track is described by a set of track parameters, η, at a particular point, it will be described by another set of parameters, $\eta' = f(\eta)$, at another point on the track. It is then easy to express the variance matrix at a new point by a simple change of variables. Exactly what is meant by describing the track at a point is discussed below.

The second part of the transport is to 'update' the track parameters and the error matrix taking into account effects from interactions with material the particle passes through. The interactions can be divided in two categories. The first is Multiple (Coulomb) Scattering, MS. This does not affect the estimated track parameters after transport, it only contributes to the error matrix. The second category is energy loss, for example due to ionization, which change the track parameters. This also affects the error matrix, not only from the effect of straggling in the energy loss, but also by a more subtle effect which occurs when the curvature is changed.

Below, we will first discuss how the transport is made in the absence of material, i.e., the simple change of variables. Then we will show how the effect of MS is accounted for and finally the effects of energy loss will be discussed.

4.1 Transport in vacuum

This part of the transport is in some sense trivial, but it is still worth describing in detail. The track, η, with variance matrix, V, is given at a point, x. The point x should be thought of as the origin of a local coordinate system in which the track is described, see Figure 4. The local coordinate system is simply a translation of the global coordinate system. We define the operation of transporting a set of track parameters as the operation of finding the new track parameters, η', with variance matrix, V', at a new reference point, x'. This formulation of the problem of track parameter transport is slightly different from what is used elsewhere, where, for example, a transport is made a certain distance along the track or to the interception of the track with some given surface. The approach taken here is, conceptually very simple and will be very useful for the implementation of the Billoir fitter.

The transport routines described here work in a uniform magnetic field along the z-axis, such that the track forms a helix in space. However, this approach can be modified to handle nonuniform magnetic fields, see Section 6.

The track parameters at the new reference point, x', can be written as

$$\eta' = f(\eta). \tag{25}$$

Defining the transport matrix, T, by

$$T_{ij} = \frac{\partial \eta_i'}{\partial \eta_j}. \tag{26}$$

This allows us to write the new error matrix as

$$V_{ij}' = \langle \delta\eta_i' \delta\eta_j' \rangle = \langle T_{ik}\delta\eta_k T_{jl}\delta\eta_l \rangle = T_{ik}V_{kl}T_{jl} \tag{27}$$

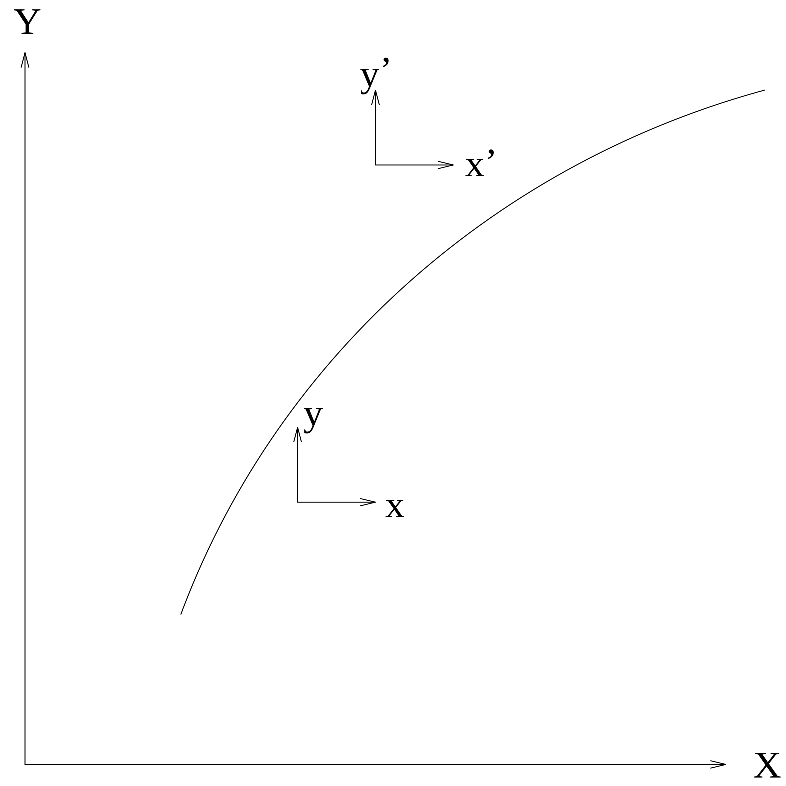

Figure 4: The different coordinate systems used in the transport. The XY coordinate system is the global coordinate system. The track parameters, η, are originally given in the xy coordinate system. In the transport operation, the track is described in the x'y' coordinate system. That is the track parameters are always given at the point of closest approach to the (local) origin in the xy-plane. When material is present, i.e., transport is no longer in vacuum, the transported track parameters will also be modified by the effects of the material, and not only the "geometrical" effect of describing the same circle at different points.

or, in matrix form, simply

$$V' = TVT^t \tag{28}$$

The above lines of algebra are implemented in the routine `t_to_t`. The transport is exact in the sense that there are no approximations made about the transport distance being small compared to the radius of curvature. Note also that there are only 10 nontrivial elements in T, see Appendix C. This is explicitly used in the multiplication of TVT^T. This routine also makes a linear transport of track parameters, used by the linearized track fit. For further discussion, see Section 2.3 about starting the Billoir fit. The linearized track parameters are transported by

$$\Delta\eta' = T\Delta\eta. \tag{29}$$

Again we emphasis that the transport in vacuum is the description of the same track at different points. The problem is simply a question of geometry.

4.2 Multiple scattering

The treatment of multiple scattering makes the assumption that the scattering occurs in a thin plane at the point of closet approach of the track to the new reference point. The scattering is assumed to be gaussian with a width of θ_0 in the scattering plane. θ_0 is calculated with routines described in Section 11.2. It is obvious that the scattering will affect the variance matrix elements $V_{\phi\phi}$ and V_{tt}, which describe the direction of the track, as MS adds an uncertainty in the direction. However, it does not change the momentum of the track. As the total momentum of the track depends on t and K, there will also be contributions from MS to the variance matrix elements V_{KK} and $V_{\mathrm{K}t}$. The rest of this section shows this in detail.

Figure 5 shows the the geometry needed to understand how multiple scattering effects the track parameters. The vector, v, is a unit vector pointing along the direction of the track. The two angles, ϕ and λ, describe the direction of the track. ϕ is simply one of the track parameters, while λ is related to t by $t = \tan\lambda$. Multiple scattering is described by two uncorrelated scatterings of θ_0 in two perpendicular planes which contain the vector v, see [2]. As the first plane, we chose the plane spanned by the vectors v and $\hat{z}$. It is obvious that a scattering in this plane only effects $V_{\lambda\lambda}$ [3]

$$V_{\lambda\lambda} \rightarrow V_{\lambda\lambda} + \theta_0^2 \tag{30}$$

[3] Given an angle ψ with a variance matrix $V_{\phi\phi}$, $V_{\phi\phi} = \langle(\psi - \bar{\psi})^2\rangle$. If this angle is subject to an change of direction by an angle θ, such that the mean is zero and the rms is θ_0, like in the case of

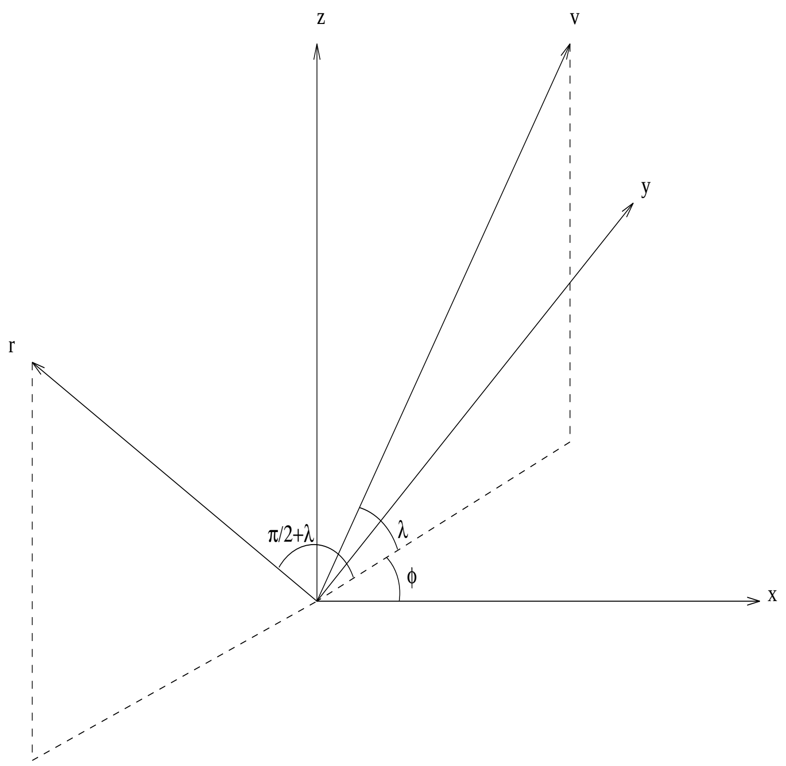

Figure 5: Geometry of multiple scattering. The direction of the track is indicated by the direction vector v in the figure. The multiple scattering is described as two independent scatterings in two perpendicular planes that both contains the vector v. As the first plane we take the plane that is spanned by v and the z axis. The second plane is taken as the plane that is perpendicular to the vector r in the figure. r lies in the first plane and is perpendicular to v.

or expressed in terms of the track parameter t,

$$V_{tt} \to V_{tt} + (1+t^2)^2\theta_0^2. \tag{31}$$

The other scatter plane can be described as the plane perpendicular to the vector r, see Figure 5. A rotation of an angle θ_0 around r corresponds to a scattering θ_0 in the second plane. The goal is to find how this affects ϕ (λ is, as will be shown below, unchanged under the infinitesimal rotation considered). One way of obtaining the desired result is to note that under a rotation, θ_0, around the r axis $v \to v + dv$ where

$$dv = \theta_0 v \times r. \tag{32}$$

Using

$$v = (\cos\phi\cos\lambda, \sin\phi\cos\lambda, \sin\lambda) \tag{33}$$

$$r = (-\cos\phi\sin\lambda, -\sin\phi, \sin\lambda, \cos\lambda) \tag{34}$$

we find

$$dv = \theta_0(-\sin\phi, \cos\phi, 0). \tag{35}$$

From this we can deduce

$$\cos\phi \to \cos\phi - \frac{\theta_0}{\cos\lambda}\sin\phi = \cos(\phi + \frac{\theta_0}{\cos\lambda}) \tag{36}$$

which implies that

$$V_{\phi\phi} \to V_{\phi\phi} + \frac{\theta_0^2}{\cos^2\lambda} = V_{\phi\phi} + \theta_0^2(1+t^2). \tag{37}$$

This is the first part of the treatment of multiple scattering. There is another effect to be considered, stemming from the fact that the momentum, or more importantly the error in the momentum, is not affected by multiple scattering. To see how this comes about consider the momentum as a function of the track parameters

$$P = \frac{A}{|\mathrm{K}|}\sqrt{1+t^2}. \tag{38}$$

Differentiating this gives

$$dP = -d\mathrm{K}\frac{P}{\mathrm{K}} + dt\frac{Pt}{1+t^2}. \tag{39}$$

MS, then with $\phi' = \phi + \theta$ it is easy to see that $V_{\phi'\phi'} = V_{\phi\phi} + \theta_0^2$ if the θ, the MS, is uncorrelated with ϕ.

Momentum is conserved under MS, meaning $dP = 0$, which gives

$$dK = dt \frac{Kt}{1+t^2}. \tag{40}$$

From this it is easy to see that the following elements in the variance matrix need to be updated

$$V_{Kt} \rightarrow V_{Kt} + Kt(1+t^2)\theta_0^2, \tag{41}$$

$$V_{KK} \rightarrow V_{KK} + K^2 t^2 \theta_0^2. \tag{42}$$

To summarize, there are 4 elements, $V_{\phi\phi}$, V_{tt}, V_{Kt}, and V_{KK} that must be updated due to the uncertainty caused by multiple scattering. This is done in the routine `klmn_trans`.

4.3 Energy loss

When the particle passes through material it loses some amount of energy, ΔE, which affects its trajectory. As there is straggling in the energy loss, it is not deterministic. Let δE denote the r.m.s. of the straggling in ΔE. The first thing to do is to update the curvature, K, with the effect of the energy loss, which can be written in the following way

$$K' = K \frac{P}{\sqrt{P^2 + 2E\Delta E + \Delta E^2}}. \tag{43}$$

The other effect that we need to incorporate is the straggling. This will add a contribution to the variance matrix element V_{KK}. Using

$$|K| = \frac{A\sqrt{1+t^2}}{\sqrt{E^2+m^2}} \tag{44}$$

it is easy to find that

$$V_{KK} \rightarrow V_{KK} + \frac{K^2 E^2}{P^4} \delta E^2. \tag{45}$$

However, there is yet one more effect of the change of curvature that is important for low momentum tracks. When a track changes its curvature the variance matrix is also affected. It is a geometrical effect and is not related to the effects of straggling. To work this out consider the energy loss as a change of variables

$$\eta \xrightarrow{\Delta E} \eta' \tag{46}$$

where the primed and unprimed variables are related by

$$\mathrm{K}' = \mathrm{K}\frac{P}{\sqrt{P^2 + 2E\Delta E + \Delta E^2}}, \tag{47}$$

$$\phi' = \phi, \tag{48}$$

$$d' = d, \tag{49}$$

$$t' = t, \tag{50}$$

$$z' = z. \tag{51}$$

There are only two non-trivial derivatives,

$$a \equiv \frac{\partial \mathrm{K}'}{\partial \mathrm{K}} = \frac{\mathrm{K}'^3}{\mathrm{K}^3}\frac{E'}{E} \tag{52}$$

$$b \equiv \frac{\partial \mathrm{K}'}{\partial t} = \frac{t}{1+t^2}\mathrm{K}'(1 - \frac{\mathrm{K}'^2}{\mathrm{K}^2}\frac{E'}{E}). \tag{53}$$

This gives

$$T = \frac{\partial \eta'}{\partial \eta} = \begin{bmatrix} a & 0 & 0 & b & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{54}$$

The new variance matrix, V', is given by $V' = TVT^T$, differs from V in elements

$$V'_{\mathrm{KK}} = a^2 V_{\mathrm{KK}} + 2ab V_{\mathrm{K}t} + b^2 V_{tt}, \tag{55}$$

$$V'_{\mathrm{K}\phi} = a V_{\mathrm{K}\phi} + b V_{t\phi}, \tag{56}$$

$$V'_{\mathrm{K}d} = a V_{\mathrm{K}d} + b V_{td}, \tag{57}$$

$$V'_{\mathrm{K}t} = a V_{\mathrm{K}t} + b V_{tt}, \tag{58}$$

$$V'_{\mathrm{K}z} = a V_{\mathrm{K}z} + b V_{tz}. \tag{59}$$

For an inward track fit, energy loss is added on, with low momentum, where 10% of the momentum is lost $\mathrm{K}'/\mathrm{K} = 0.9$ and $V'_{\mathrm{KK}} = 0.9^6 V_{\mathrm{KK}} = 0.53 V_{\mathrm{KK}}$. If this effect is ignored the error on the curvature would be overestimated.

5 Adding information to the fit

In Section 2.2 the addition of information, hits, to the track fit was described. It was assumed that the hit provides a measurement, d, of some quantity, e.g., a drift distance or position on a silicon wafer. The addition of this information was done in a slightly abstract form in which the derivative, D, see Eq. 4, was needed. The purpose of this Section is to give the explicit form of the derivatives, D, for the different types of hits that are considered.

5.1 Axial wires

When treating axial wires, the derivatives needed are trivial, as it is assumed that the track parameters are expressed in a coordinate system where the wire is at the origin. Therefore the projected drift distance from the track, $d^{\min}(\eta)$, is d. This leads to only one non-zero derivative, which is trivial,

$$\frac{\partial d^{\min}}{\partial d} = 1. \tag{60}$$

5.2 Stereo wires

The stereo wire is more involved then the axial wire. The first problem is that we have to find the point of closest approach between the track and the wire. The solution to this problem can not be obtained in closed form as it involves a transcendental equation. Instead the solution is found numerically by an iterative process, which converges in few iterations with a reasonable initial guess.

The stereo wires are parameterized by s in the following way

$$x_w = x_{0w} + c_x s, \tag{61}$$
$$y_w = y_{0w} + c_y s, \tag{62}$$
$$z_w = s, \tag{63}$$

such that the vector $(c_x, c_y, 1)$ is along the direction of the stereo wire and x_{0w} and y_{0w} are the coordinates of the stereo wire in the $z = 0$ plane. The track parameterization was shown earlier but is reproduced below since it will be used. The track is parameterized by ψ

$$x_t = -d \sin\phi + \frac{1}{2\mathrm{K}} \left(\sin(\phi + q\psi) - \sin\phi \right), \tag{64}$$

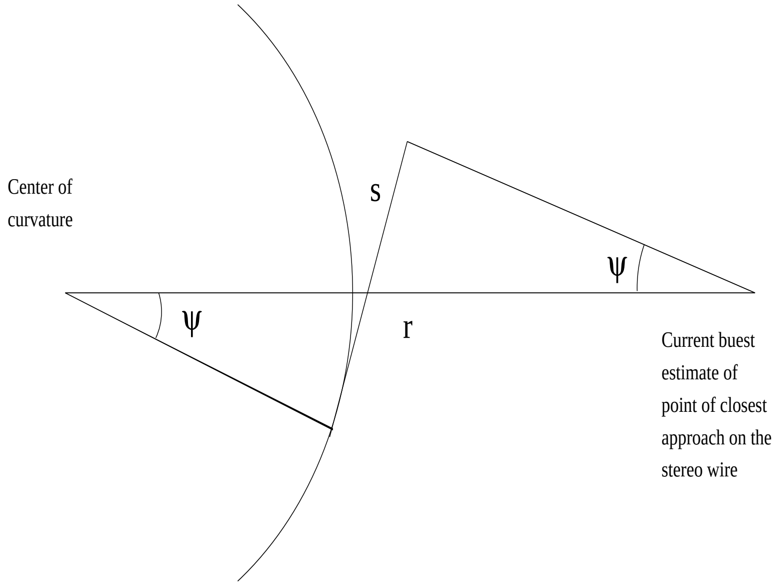

Figure 6: This figure shows how the track, the arc, is approximated with a straight line and how the distance s is found. Note that when the point is close to the track compared to the radius the distance r is well approximated by the radius of curvature. However when the point is far away, for example outside the arc as in the figure this is not a good approximation. Under normal circumstances the point, a given wire, is close to the track since the drift distance is much smaller then the curvature.

$$y_t = d\cos\phi + \frac{1}{2\mathrm{K}}\left(\cos\phi - \cos(\phi + q\psi)\right), \tag{65}$$

$$z_t = z + \frac{1}{2\mathrm{K}} t q \psi. \tag{66}$$

To iterate this the track is approximated as a straight line at the point (x_t, y_t, z_t). Using vector notation, the wire and the track are parameterized as

$$x_w = x_{0w} + v_w s_w, \tag{67}$$

$$x_t = x_{0t} + v_t s_t, \tag{68}$$

where $x_{0t} = (x_t, y_t, z_t)$ is the current best guess of the closest point on the track and

$$v_w = (c_x, c_y, 1)/\sqrt{c_x^2 + c_y^2 + 1}, \tag{69}$$

$$v_t = (\cos(\phi + q\psi), \sin(\phi + q\psi), t)/\sqrt{1 + t^2}. \tag{70}$$

We are now interested in finding the s_t and s_w which minimize the distance

$$d = \sqrt{(x_w - x_t)^2} = \sqrt{(x_{0w} + v_w s_w - x_{0t} - v_t s_t)^2}. \tag{71}$$

Solving the above equation for the s_t which minimizes d gives

$$s_t = \frac{(x_{0w} - x_{0t})(v_w(v_w \cdot v_t) - v_t)}{(v_w \cdot v_t)^2 - 1}, \tag{72}$$

where we have made use of the fact that v_w and v_t are unit vectors. This gives us a distance, s_t, to move along the track to find the new estimate of the point on the track closest to the wire, see Fig. 6.

$$\psi \to \psi + \frac{s_t}{r\sqrt{1 + t^2}}. \tag{73}$$

With this new estimate of ψ, the new best estimate of the point of closest approach of the track to the stereo wire, the steps above are repeated untill the procedure has converged. The convergence criteria is that the estimated distance to the point of closest approach is less than 10 μm. This is done in the routine `stereo_dist`.

After we have found the point of closest approach, expressed as ψ, we can now, quite easily, evaluate the derivative of the distance of closest approach to the stereo wire with respect to the track parameters, without making any approximations. It is convenient to work out the derivatives with respect to $d^2_{\rm min}$ and use

$$\frac{dd_{\rm min}}{d\eta} = \frac{1}{2d}\frac{dd^2_{\rm min}}{d\eta}. \tag{74}$$

In the next step we will be very explicit, and careful, to see how the derivatives work out. The end result is just what's expected. First we write down $d^2_{\rm min}$ with all dependencies on η:

$$\begin{aligned} d^2_{\rm min} &= \left(x_t\left(\eta,\psi_{\rm min}(\eta)\right) - x_{0w} - c_x s_{\rm min}(\eta)\right)^2 && (75)\\ &+ \left(y_t\left(\eta,\psi_{\rm min}(\eta)\right) - y_{0w} - c_y s_{\rm min}(\eta)\right)^2 && (76)\\ &+ \left(z_t\left(\eta,\psi_{\rm min}(\eta)\right) - s_{\rm min}(\eta)\right)^2 && (77)\\ &= \Delta x^2 + \Delta y^2 + \Delta z^2. && (78) \end{aligned}$$

Differentiating this with respect to η gives

$$\begin{aligned} \frac{dd^2_{\rm min}}{d\eta} &= 2\Delta x\left(\frac{\partial x_t}{\partial\eta} + \frac{\partial x_t}{\partial\psi_{\rm min}}\frac{d\psi_{\rm min}}{d\eta} - c_x\frac{ds_{\rm min}}{d\eta}\right) && (79)\\ &+ 2\Delta y\left(\frac{\partial y_t}{\partial\eta} + \frac{\partial y_t}{\partial\psi_{\rm min}}\frac{d\psi_{\rm min}}{d\eta} - c_y\frac{ds_{\rm min}}{d\eta}\right) && (80)\\ &+ 2\Delta z\left(\frac{\partial z_t}{\partial\eta} + \frac{\partial z_t}{\partial\psi_{\rm min}}\frac{d\psi_{\rm min}}{d\eta} - \frac{ds_{\rm min}}{d\eta}\right) && (81)\\ &= 2\Delta x\frac{\partial x_t}{\partial\eta} + 2\Delta x\frac{\partial x_t}{\partial\eta} + 2\Delta x\frac{\partial x_t}{\partial\eta}. && (82) \end{aligned}$$

The condition for the minimum is

$$\begin{aligned} 0 &= \Delta x\frac{\partial x_t}{\partial\psi_{\rm min}} + \Delta y\frac{\partial y_t}{\partial\psi_{\rm min}} + \Delta z\frac{\partial z_t}{\partial\psi_{\rm min}}, && (83)\\ 0 &= \Delta x c_x + \Delta y c_y + \Delta z. && (84) \end{aligned}$$

The needed derivatives, e.g., $\partial x_t/\partial\eta$ are easily computed from Eq. 65-66. This is coded in the routine `cleostereoder`.

5.3 Cathodes

This is, at least for the time being, a trivial derivative. This is mostly so because I'm not really sure what is the right thing. $dd/dz = 1$ and all other derivatives are zero.

5.4 Charge division hits

Currently treated in exactly the same way as the cathodes.

5.5 Silicon strip hits

The calculation of the derivatives for a silicon strip measurement is similar to that for a stereo wire. It proceeds in two steps. The first is to find the point of intercept of the track with the wafer, done with an iterative process very similar to finding the distance of closest approach to a stereo wire. Once this point is found, it is easy to write down, exact expressions for the derivative with respect to the track parameters. No assumptions are made about the wafers being parallel to the z-axis.

The iteration for finding the intercept, described in terms of ψ_0, is done by approximating the track as straight line. With the notation in Figure 7, the condition for the intercept with the plane of the wafer is

$$n \cdot (x_0 - (x_t + sv)) = 0. \tag{85}$$

The plane of the wafer is described by its normal, n, and a point in the plane, x_0. x_t is the current guess of the position on the track for the intercept of the track and the wafer and v is the tangent of the track at that point. It is easy to solve this equation for the distance, s, along the track to the intercept,

$$s = \frac{n \cdot (x_o - x_t)}{n \cdot v}. \tag{86}$$

This leads to the following change in ψ

$$\psi \to \psi + \frac{2s\mathrm{K}}{\sqrt{1+t^2}}. \tag{87}$$

This procedure is repeated untill the intercept is found with sufficient precision.

After the intercept of the track and the wafer is found in terms of the turning angle, ψ, the derivatives needed are easily evaluated. With u being a unit vector in the silicon wafer plane along the axis where the measurement, d, is made, the predicted distance d_t is easily found as

$$d_t = u \cdot (x_0 - x_t). \tag{88}$$

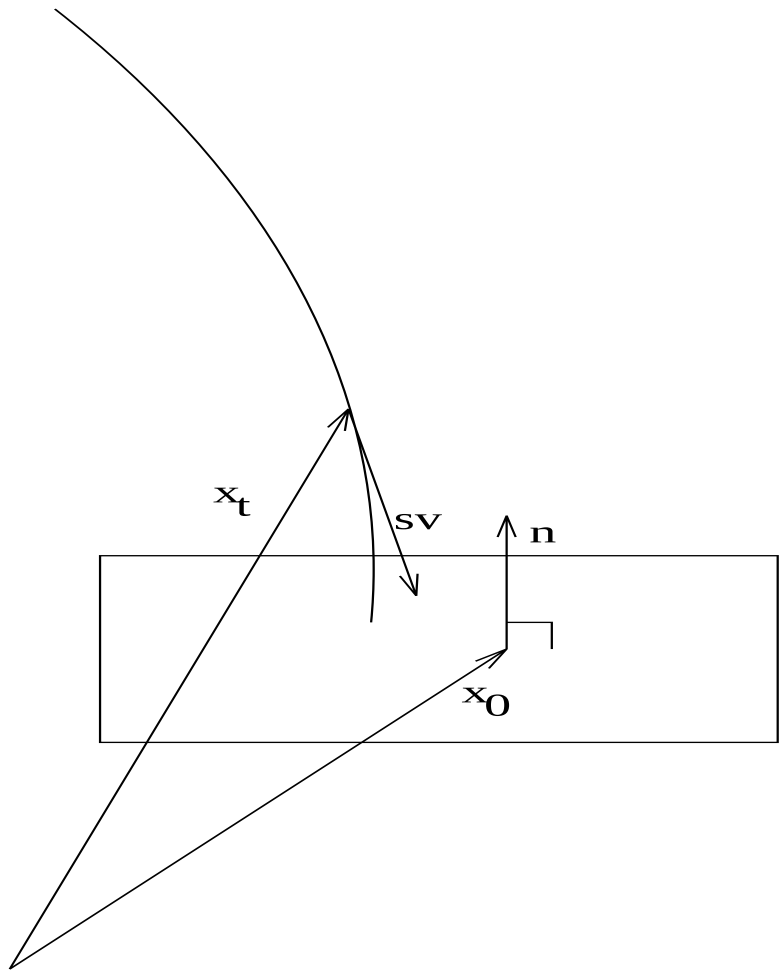

Figure 7: This figure shows how the track, the arc, is approximated with a straight line and how the distance s is to the plane is found.

The derivatives that we need are

$$\frac{dd_t}{d\eta} = u \cdot \frac{dx_t}{d\eta} = u \cdot \left(\frac{\partial x_t}{\partial \eta} + \frac{d\psi_0}{d\eta} \frac{\partial x_t}{\partial \psi_0} \right). \tag{89}$$

The only derivative here that is somewhat tricky is $d\psi_o/d\eta$. The easiest way (I know of) is to use

$$0 = n \cdot \frac{dx_t}{d\eta}(\eta, \psi_0(\eta)) = n \cdot \frac{\partial x_t}{\partial \eta} + \frac{d\psi_0}{d\eta} n \cdot \frac{\partial x_t}{\partial \eta}. \tag{90}$$

This gives

$$\frac{d\psi_0}{d\eta} = -\frac{n \cdot \dfrac{\partial x_t}{\partial \eta}}{n \cdot \dfrac{\partial x_t}{\partial \psi_0}}. \tag{91}$$

All other derivatives are trivial to evaluate. This is implemented in `cleosvxder`.

6 Non uniform magnetic field

So far the discussion has been concerned with fits of tracks in a uniform magnetic field along the z-axis. This is what the code was designed to do. However, there is a mechanism to allow for non-uniform magnetic fields. The idea is to treat the difference in the magnetic field,

$$\Delta B = B_{\mathrm{true}} - B_{\mathrm{uniform}}, \tag{92}$$

as a perturbation. The fit is essentially performed as if the field was uniform along the z-axis, and the residual magnetic field is used to correct the transport. This will be explained carefully below. One effect this has is that when you extract the momentum of the particle, say at the origin, you use the uniform magnetic field, not the actual magnetic field, i.e., the track parameter K is the curvature of the track with the current momentum in a magnetic field with the uniform strength not the local field. This might seem somewhat counter intuitive at first but is very straight forward to implement.

Starting from the well known formula for the force on a charged particle, with charge q (in unit charges), in a magnetic field B (Tesla)

$$F = 0.3 q B \times v \tag{93}$$

where v is the particles velocity in m/s and the force is given in units of GeV/s. A very useful expression is obtained after integrating the above equation with respect to time

$$P = 0.3qB \times x. \tag{94}$$

This equation describes how the momentum of a particle is changing as it goes a distance x in a magnetic field B. The equation is meaningful as long as the distance, x, in the magnetic field can be approximated with a straight line. Note that a global form of this is the familiar expression $P = 0.3qB\rho$ which gives the (transverse) momentum of a particle with a radius of curvature ρ.

Here we are interested in calculating the residual effect of a magnetic field ΔB as a particle goes a distance x. Remember that the routine `t_to_t` does the transport assuming that the magnetic field is uniform. It is interesting to note that `t_to_t` does not need to know the magnetic field, it is a purely geometrical operation.

The residual magnetic field gives the following change in the momentum of the particle:

$$\Delta P = 0.3q\Delta B \times x. \tag{95}$$

The next step is to correct the track parameters for this change. The track parameters affected by this are K, ϕ, and t. We are interested in finding ΔK, $\Delta\phi$, and Δt to correct for the non uniformity in the magnetic field. We write down the changes of a particle's momentum with respect to the changes in the track parameters and equate them with the the calculated changes from the the residual magnetic field.

$$\Delta P_x = -\frac{\Delta K}{|K|}P_x - \Delta\phi P_y, \tag{96}$$

$$\Delta P_y = -\frac{\Delta K}{|K|}P_y + \Delta\phi P_x, \tag{97}$$

$$\Delta P_z = \frac{\Delta K}{|K|}P_z + \Delta t\frac{A}{|K|}. \tag{98}$$

Solving these equations yields

$$\Delta\phi = \frac{|K|}{A}(\Delta P_y \cos\phi - \Delta P_x \sin\phi), \tag{99}$$

$$\Delta K = \frac{|K|K}{A}(\Delta P_x \cos\phi + \Delta P_y \sin\phi), \tag{100}$$

$$\Delta t = \Delta P_z\frac{|K|}{A} + \frac{t\Delta K}{K}. \tag{101}$$

This is easiest to derive using the constraint $\Delta P \cdot P = 0$. The track, is then updated with these corrections. The magnetic field, ΔB, is evaluated at the mid point of the transport step. One question remains that I have not yet been able to answer precisely; what is the small parameter? The obvious candidate $\Delta B/B$ is not right since the reliability in this method obviously depends on the length of the transport step. One other candidate is $|\Delta P|/|P|$, this parameter is small both when ΔB and the step length is small.

Again we remind you that the extracted track parameters are to be interpreted as track parameters for a track in the uniform magnetic field along the z axis, not in the local magnetic field.

This is implemented in `klmn_trans` but is commented out. The user has to provide a routine that returns the magnetic field. In the commented out code this routine is called `gufld`.

7 Smoothing

After a track fit we have obtained the optimal track parameters at the point of closest approach to the origin. However, we have a very poor estimate of the track parameters at any other point along the track. As the fit is performed towards the innermost point we successively obtained the best estimate of the track parameters *based on the hits so far encountered.* If we want to obtain the optimal track parameters at any point on the track based on all hits on the track we have to run the fit from both ends of the track to a given point and average the result of the two fits. This procedure is know as smoothing [6]. A practical way of doing this is to save the track parameters from the inward pass of the fit and then on the second, outward, pass, average the estimates, which now is based on the hit inside the point, with the first estimate at that point from the inward pass. It is important to realize that these two estimates are uncorrelated even in the presence of multiple scattering.

Suppose we are given to estimates, η_1 and η_2, for the same quantity η with the corresponding variance matrices V_1 and V_2. η_1 and η_2 are assumed to be uncorrelated. We now want to find the best estimate of η. We write down the likelihood as a function of η

$$\mathcal{L} = e^{-\frac{1}{2}(\eta-\eta_1)^t V_1^{-1}(\eta-\eta_1) - \frac{1}{2}(\eta-\eta_2)^t V_2^{-1}(\eta-\eta_2)}. \tag{102}$$

Now we want to find the maximum of the likelihood, so take a derivative with respect to η^t and set it to zero and solve for η

$$0 = \frac{\partial(-ln\mathcal{L})}{\partial\eta^t} = V_1^{-1}(\eta - \eta_1) + V_2^{-1}(\eta - \eta_2). \tag{103}$$

Solving this for η gives

$$\eta = (V_1^{-1} + V_2^{-1})^{-1}(V_1^{-1}\eta_1 + V_2^{-1}\eta_2). \tag{104}$$

As always, we also want to know what the variance matrix is for η, this is found simply from the definition

$$V_\eta = \langle\delta\eta\delta\eta^t\rangle = ... = (V_1^{-1} + V_2^{-1})^{-1}. \tag{105}$$

We can also rewrite Eq. 104 in the following way

$$\eta = V_2(V_1 + V_2)^{-1}\eta_1 + V_1(V_1 + V_2)^{-1}\eta_2. \tag{106}$$

This shows that only $V_1 + V_2$ needs to be invertible to allow for the construction of the average.

8 Residuals

This section is devoted to a discussion of the residuals calculated in the fit and their distribution, i.e., their errors. This is important since if the residual distributions are not understood, the derived quantities, like the χ^2-distribution and the variance matrix of the track parameters, are also not understood.

The fitter can calculate two types of hit residuals; where the hit included in the track fit, and without the hit included. First, we discuss the case when the hit is *not* included in the fit, as this is the simple case. The residual with the hit included is somewhat more complicated, though the final result is very simple.

When the hit is not included in the fit, the projected measurement, d_{track}, from the track is uncorrelated with the measurement, d_{meas}, and the residual,

$$r = d_{\mathrm{meas}} - d_{\mathrm{track}}, \tag{107}$$

have the variance

$$\sigma_r^2 = \sigma_{d_{\mathrm{meas}}}^2 + \sigma_{d_{\mathrm{track}}}^2. \tag{108}$$

When the hit is included in the track fit the residual and the projected measurement from the track becomes correlated and Eq. 108 is no longer correct. The easiest way, as far as I know, to find the variance is to first combine the measurement, d_{meas}, with the projected track without the hit include , d_{track}. Averaging these uncorrelated estimates give

$$d_{\mathrm{c}} = (\frac{d_{\mathrm{track}}}{\sigma^2_{d_{\mathrm{track}}}} + \frac{d_{\mathrm{meas}}}{\sigma^2_{d_{\mathrm{meas}}}})\sigma^2_{\mathrm{c}}, \tag{109}$$

where

$$\frac{1}{\sigma^2_{\mathrm{c}}} = \frac{1}{\sigma^2_{d_{\mathrm{track}}}} + \frac{1}{\sigma^2_{d_{\mathrm{meas}}}}. \tag{110}$$

σ_{c} is the variance of d_{c}, the estimated projected drift distance for a hit where the hit is included in the track fit. Defining the residual r by

$$r \equiv d_{\mathrm{meas}} - d_{\mathrm{c}} = d_{\mathrm{meas}} - (\frac{d_{\mathrm{track}}}{\sigma^2_{d_{\mathrm{track}}}} + \frac{d_{\mathrm{meas}}}{\sigma^2_{d_{\mathrm{meas}}}})\sigma^2_{\mathrm{c}}, \tag{111}$$

it is now easy to find the variance on the residual

$$\sigma^2_r = \langle \delta r \delta r \rangle = ... = \sigma^2_{d_{\mathrm{meas}}} - \sigma^2_{\mathrm{c}}. \tag{112}$$

This shows that the variance of the hit residuals are simply given by the variance of the resolution of the measurement minus the variance of the projected track, including the hit. Note that the difference between this case and the case were the hit is left out of the tack fit is just the sign. σ^2_r is always positive since σ^2_{c} contains the information from the hit, and therefore is smaller than $\sigma^2_{d_{\mathrm{meas}}}$.

9 User interface

This section will describe the user interface, i.e., the calling sequence that is needed to perform a track fit, or to use the utility routines for track parameter transport, including the needed initialization. One of the main concepts of this fitter is that it is stand-alone, i.e., it only operates on data in its own structures and does not use any information from the structures in `DUET`. The user therefore has to fill the the structures with hit information prior to calling the fitter.

In Section 9.1 the initialization of the geometry and material properties is described. This is done through a call to the routine `klmn_get_geom`. This routine is called once per job and it builds the structures containing the volumes through which tracks are traced. It also extracts tables for the calculation of energy loss and multiple scattering from GEANT.

Information about the hits is detailed in Section 9.2. Here the details about the format of the hit information are specified, other relevant things, such as the ordering are discussed. All information about the hit is independent of any `DUET` structures, i.e., the information is specified as geometric positions etc., and not wire numbers.

The fitter also needs a seed. This is discussed in Section 9.3. This section contains discussion about how you might use the Kalman filter to continue a previous fit, i.e., how you start with a non-empty variance matrix, as well as other options for starting and ending the fit, e.g., where the track parameters are reported.

After filling the relevant input information, the user calls the routine **`klmn_fit`** which then performs a fit to the data. The routine **`klmn_fit`** takes 4 arguments, the first of which is what particle hypothesis to do the fit for, 0 means a fit to all 5 particle hypothesis. The second argument is whether the smoothing should be performed or not, the third is to tell the fit if it should calculate residuals or not. The last argument is to specify if the hit should be included in the fit when the residual is calculated. Note that using the Kalman filter makes it very easy to calculate the hit residuals with or without including the hit.

In Section 9.4, the various outputs which are filled are described. Here we have information about the track parameters for different particle hypothesis, error flags that might have been set to indicate that the fit was ill behaved, χ^2, the number of degrees of freedom, hit residuals, arc length, and other things.

9.1 Initialization

Before a track can be fitted using the Billoir fitter, the material structures must be initialized. There are two independent parts in this initialization. First is the initialization of the volumes, and their positions. This is a hierarchical structure which captures the volumes properties of being inside each other. Here it is appropriate to point out that `GEANT` allows for volumes to be declared as "MANY", which means that volumes are allowed to overlap on the same level in the decay tree. This is not supported by the fitter. To guarantee correct results, the volumes used by the fitter should be "ONLY" volumes in `GEANT`.

The second part is to initialize the material properties. After initialization of the volumes, the needed material properties for calculation of energy loss and multiple scattering are stored. This information is stored in the common block `mater_prop`. Associated with each volume is a pointer to the corresponding volume number in the `mater_prop` common.

9.1.1 Volume structure initialization

The volume structure is divided into two parts. The first, `chamber.inc`, contains a list of the volumes beside the volumes inside the precision tracking device. The volumes in the precision tracking device are described in `wafer.inc`. The idea of this separation is that the volumes in these commons are slightly different. The volumes in `wafer.inc` are typically smaller volumes which can be tabulated to get faster access to the right volumes inside a complicated volume structure; while volumes in the `chamber.inc` are typically larger volumes, such as drift chambers, and do not allow for a convenient tabulation.

The volumes used are specified in an ASCII file. Again, there are two such files, one for the volumes in `chamber.inc` and one for the volumes in `wafer.inc`. These files are stored in runfil and are named material_vddr_ptl.dat, mater_vddr_svx.dat, mater_ptl.dat, and mater_svx.dat. The "_ptl" or "_svx" files are to be used with the PTL or SVX configuration respectively. Which file is used is determined based on the run number. A sample of such a file for the volumes in `chamber.inc` is given below:

```
1
CLEO
2
CLEO 1 CBRO 1 1
2
CLEO 1 CBRI 1 1
3
CLEO 1 CBRO 1 DRII 1 1
4
CLEO 1 CBRO 1 DRII 1 DRSK 1 1
4
CLEO 1 CBRO 1 DRII 1 DRSG 1 1
4
CLEO 1 CBRO 1 DRII 1 DRC6 1 1
...
```

```
4
CLEO 1 CBRI 1 PTL  1 PTL- 1 1
4
CLEO 1 CBRI 1 BPVO 1 BPBE 1 1
4
CLEO 1 CBRI 1 BPVO 1 BPAG 1 1
0
```

There are a few rules for how to construct this file:

- Each volume element is specified on *two* lines. the first line is a single integer specifying the depth in the volume tree at which the volume is located. A zero (0) here indicates the end of this file.
- Before a volume can be read in its parent must have been read in.
- The daughters of one volume must be *consecutive*.
- The mother volume of the precision tracking volume should be included in the `chamber.inc` list.

This geometry initialization is done by calling the routine `klmn_get_geom`. This routine reads in the material description, first to the common block `chamber.inc` and then to the common block`wafer.inc`, then it proceeds to read the material properties needed for the volumes which have been read.

9.1.2 Initialization of material properties

After the volumes have been read in the routine `klmn_get_geom` goes through the different volume types as well as the different materials, and extracts the information needed for the volumes to be used. For the volumes, it extracts the type of geometry parameters needed to describe the volume. For the material properties, it extracts the energy loss tables and the multiple scattering properties.

9.1.3 Magnetic field initialization

The last thing that needs to be initialized is the magnetic field, also done in the routine `klmn_get_geom`. The magnetic field is assumed to be uniform along the z axis. There is, however, a mechanism for dealing with non-uniform magnetic fields, this is described in Section 6.

9.1.4 Alternative initialization

There is an alternative way of doing this initialization, done by a call to the routine `klmn_read_geom`. This routine reads an ASCII file which contains all of the information that `klmn_get_geom` extracted from the `GEANT` structures. This is convenient to use in a stand alone fitter or in a `DRIVER` job, where there is no easy access to the `GEANT` commons. In a `DRIVER` job, you might be interested in doing track transport with the utility routines which have been developed for the fitter.

9.2 Hit information

The hit information is stored in the common block `cleofitcom`. This common block contains a series of variables, see Table 1, which specify the hits. The sections below detail the different hit types supported by the fitter and what the exact meaning of the different variables are.

Variable	Meaning
`itype`	Type of hit
`ttype`	Transport type
`x_0`	x-coordinate of the hit.
`y_0`	y-coordinate of the hit.
`d`	The measurement, particle hypothesis dependent.
`sigma`	The precision of the measurement, particle hypothesis dependent.
`cx`	The directional cosine of the stereo wires in x
`cy`	The directional cosine of the stereo wires in y
`w_org`	A point on a silicon wafer
`u_hat`	1:st normal on silicon wafer
`v_hat`	2:nd normal on silicon wafer

Table 1: The different variables used for specifying the hits. For exact description of how these variables are used for the different types of hits supported by the fitter see Sections 9.2.1-9.2.5 that details the different hit types.

The hits must be ordered as the fitter does not do any ordering. The hits are ordered according to their arc length from the interaction point. The hit furthest from the IP is hit number one, and the hit closest to the IP is hit number `nhits`, which also must be filled before the call to `klmn_fit`.

Below, each different type of measurement which the fitter handles is described, as well as how the different variables in the common `cleofitcom.inc` are used.

9.2.1 No hit

This has `itype`=0, and should normally not be filled by the user. It is used internally in the fit when the fitter is run in hit deletion mode to mark the hits which are deleted. Only the contents in the variables `x_0` and `y_0` are used. They are used for transport, i.e., the track is transported to this point. `ttype` should be set to zero, ensuring that transport is performed from this point to the next point (inwards).

9.2.2 Axial wires

This has `itype`=1, and `x_0` and `y_0` are the positions of the wire. These coordinates are also used for the transport, i.e., they define the reference point to transport. `d` is the signed drift distance, and `sigma` is the resolution of the drift distance. `ttype` should be set to zero, ensuring that transport is performed from this point to the next point (inwards).

9.2.3 Stereo wires

This has `itype`=2, and `x_0` and `y_0` are the coordinates of the stereo wire at the $z = 0$ plane. `cx` and `cy` are defined by the parameterization of the wire, $x = \texttt{x\_0} + s * \texttt{cx}$, $y = \texttt{y\_0} + s * \texttt{cy}$, and $z = s$, where s is an arbitrary parameter (equal to the z-position). Again, `d` is the signed drift distance, and `sigma` is the resolution of the drift distance. `ttype` should be set to zero, ensuring that transport is performed from this point to the next point (inwards).

9.2.4 Cathodes

This has `itype`=3, and `x_0` and `y_0` are the coordinates of the measurement, as well as the point to which the track is transported. `d` is the measured z-position and `sigma` is error on this measurement. `ttype` should be set to zero, ensuring that transport is performed from this point to the next point (inwards).

9.2.5 Silicon plane hits

This has `itype`=4. In the current implementation, silicon hits are added on one at the time. There is the option of suppressing transport between the addition of hits, done by setting `ttype`=1. This is used when adding two hits in the same silicon wafer, to avoid doing a transport parallel to the silicon plane. The first entry in the list of hits should have `ttype`=1.

`x_0` and `y_0` are the point to which the track is to be transported,which should be a point in the neighborhood of where the track hit the wafer. `w_org` are the coordinates of the wafer origin with respect to the coordinates $(\texttt{x\_0}, \texttt{y\_0}, 0)$. `u_hat` and `v_hat` are two unit vectors in the silicon wafer plane, `u_hat` being along the direction of the measurement. `d` is the measurement along `u_hat`, and `sigma` is the resolution on the measurement.

9.3 Seed information

As described earlier in this note, the fitter needs a seed which in general comes from pattern recognition. This seed is specified in `k_start`, `d0_start`, `phi_start`, `z_start`, and `t_start`. These parameters are referenced in a coordinate system with origin in `x_start`, `y_start`, and `z_start`, typically (0,0,0) except when a fit is continued. `start_type` is typically zero which means that the fit is starting from scratch. If `start_type`=1 the fit is continued, meaning that there is an initial information matrix, which is non zero. This is specified, if needed, in `v_start`. If `start_type`=0, `v_start` is never used.

Before calling the fitter, you also need to specify where the track parameters are reported. At the moment there are two option, controlled by the variable `end_type`. `end_type`=0 means that the track parameters are reported at the point of closest approach to the origin, where as `end_type`=1 reports the track parameters at the last hit. The track parameters are referenced in a coordinate system with origin at `xo_in`, `yo_in`, and `zo_in`.

9.4 Fit results

This section describes where different outputs from the track fit can be found.

9.4.1 Track parameters

the resulting track parameters and error matrices from the fit are stored in the common block `klmntrk`. The track parameters are stored in `eta_in` with the corresponding error matrix in `v_in`. If `klmn_fit` was called with hypothesis 0, then all five hypothesis will be filled unless an error occured, see Section 9.4.4. Otherwise only the hypothesis specified in the call to `klmn_fit` will be filled. The track parameters are referenced in a coordinate system with origin at (`xo_in`, `yo_in`, `zo_in`). This is $(0, 0, 0)$ if `end_type`=0, see Section 9.3, otherwise it is the coordinate of the last (innermost) hit.

The track parameters and error matrix are also reported at the outermost hit, stored in `eta_out` and `v_out`. The track parameters are referenced in a coordinate system with origin at (`xo_out`, `yo_out`, `zo_out`). For a multi particle fit, only the pion fit is saved in `eta_out`.

9.4.2 χ^2 and number degrees of freedom

The χ^2 built on the inward pass is stored in the variable `chisq_in` found in the common block `klmntrk`. The χ^2 is calculated for the different particle hypothesis used in the fit. The number of measurements is stored in `n_meas`, in the common block `cleofitcom`. The number of degrees of freedom in the fit is simply `n_meas`-5, as there are 5 track parameters. `n_meas` accounts for the number of hits deleted in the hit deletion pass.

9.4.3 Hit residuals

Another output from the fit is the individual hit residuals. These residuals are stored in the common block `klmncm`. The residuals are stored in the `res_s`, and the error (squared) of these residuals are stored in `sigma2_s`. Depending on what arguments are passed to `klmn_fit`, the hit residuals will or will not include the hit, see Section 8.

9.4.4 Error flags

The common block `klmnerror` contains the error flags which might be set during the track fit. There is an error flag for each of the 5 different particle hypothesis used in the fit. The error flags are initialized to `FALSE` at the beginning of a fit, and set to `TRUE` if an error condition is detected. An error is flagged in the variable `klmnerror_error`, and an error number is returned in `klmnerror_nr`. The type of error conditions are thus far limited. Table 2 shows the different errors which can be detected.

Error number	Meaning
1	Curvature of seed to large (|K|>14.0).
2	The track is starting outside the detector mother volume.
3	The track exited the mother volume of the detector.

Table 2: The error conditions that can be returned from the fitter.

9.4.5 Miscellaneous

Other things that the fitter can calculate include the arc length of the track out to some given radius or z-position. This is still in development and will be described when more testing and work has been done to understand it. The call to this code is commented out at the end of `klmn_fit`.

10 Program structure

The goal of this section is to explain how the code works, as well as how the different steps which make up the Billoir fit are implemented. The fitter is implemented in Fortran77, however we have not followed the CLEO convention limiting the names of variables, functions, etc. to six characters [5].

This code has been an ongoing development for a while and has changed significantly over the time. The current incarnation of the code is in reasonable shape. However, further improvements in the structure of this program seem meaningless. If any further developments are to take place, a re-write in C++ would be my way of attacking the problem.

To describe how the code works, we start out in Section 10.1 by going through what happens in a track fit. This discussion is based on a set of flow charts of the program. Then we continue, in Section 10.2, to discuss how the program is controlled. This is essentially based on a table of the different control variables which decide what the code is doing. As discussed in this section, there are two types of control flags; internal and external. External control flags are set by the user and internal flags are controlled by the program.

The discussion continues with a few of the more central routines in the track fitter. The first routine considered is `klmn_fit`, which is the main routine the user will call. This routine performs some initialization, and calls the next routine that we consider, `cleofit_lin`. This routine is the main control routine for a track fit, and contains the loop over the hits in a track fit. We then look at the routines `klmn_add_hit` and `klmn_trans`, called from `cleofit_lin`. These are the routines which perform the two steps at the heart of the Billoir fit: the addition of a hit, and the transport of track parameters. Last, we look at the routine `rad_length`, which traces the track through the detector and and finds the amount of material it goes through.

In Section 10.4, we look at the initialization that takes place before a track fit can be performed, mainly the material initialization and the magnetic field.

We conclude the discussion about implementation with a collection of tips and hints in Section 10.5, including a discussion about the problems that we have encountered, and how we solved them.

10.1 What happens in a track fit

After the hit information has been filled by the user, he calls the routine `klmn_fit`, which is the entry point to the Kalman fitting package, see Figure 8. This routine performs the initialization of the additional stopping points added, if necessary, along the track to make the transport accurate. The additional stopping points are added in the routine `stop_init`.

After initialization, the first fit is performed. It is an inward fit, followed by an outward pass. If `klmn_fit` is called with the argument `part_type`=0, the first fit is by default the pion hypothesis, otherwise the hypothesis requested in `part_type` is used. This first fit is done with the option to calculate the hit residuals, used for hit deletion. After the first fit is performed, the hit deletion routine, `del_hit`, is called. If a hit is deleted, the fit is done once more. This is repeated until there are no more hits deleted from the track. If the fitter is called with `part_type`=0 the fits for the other particle hypothesis are done.

`klmn_fit` invokes the routine `cleofit_lin` to perform a track fit. `cleofit_lin` is the main driver routine for the Billoir fit. After initialization of the track parameters and the variance matrix, the subroutine consists of a loop over the hits on the track. A hit is added to the track fit by a call to the routine `klmn_add_hit`. This routine

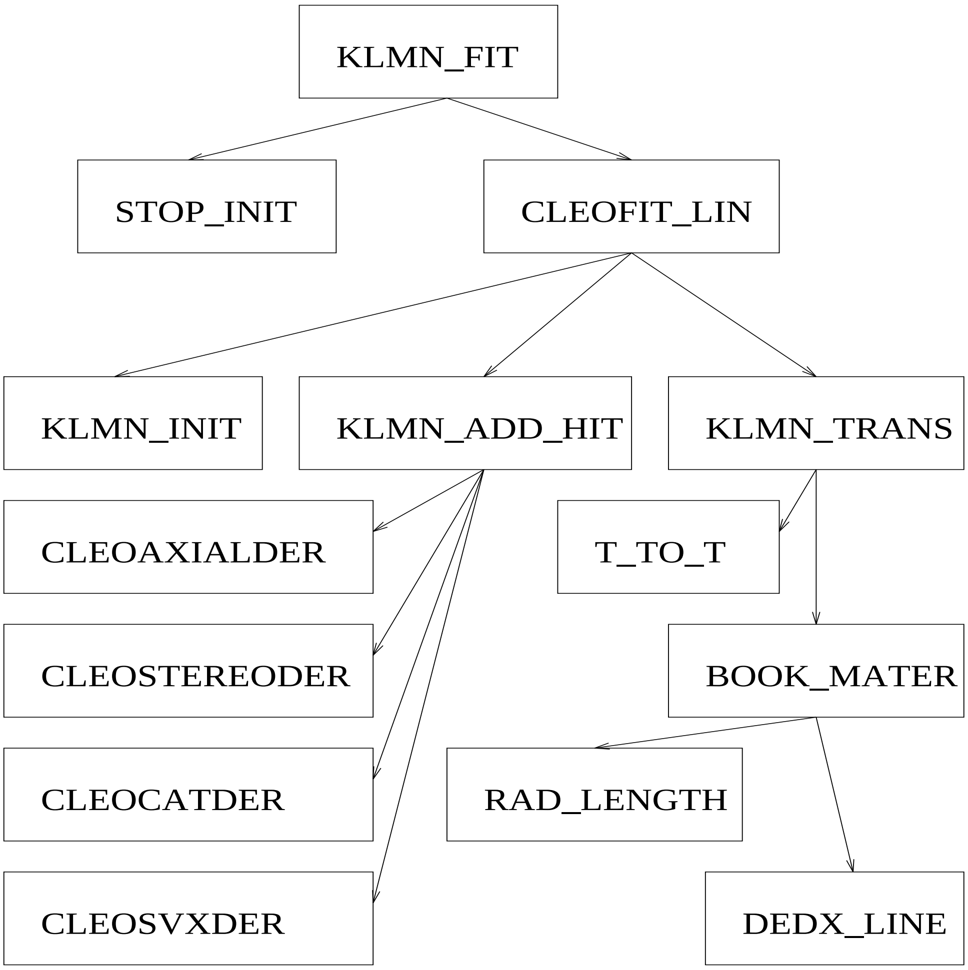

Figure 8: The flow of the program in a fit. `klmn_fit` is the main fitter routine which is invoked to perform a fit. The fit is performed by the routine `cleofit_lin`. The two routines `klmn_add_hit` and `klmn_trans` performs the operations of adding a hit to the track and transporting the track parameters and error matrices between hits.

calculates the derivatives and performs the "averaging" of the hit with the current estimate of the track parameters to obtain a new improved estimate of the track parameters, also building the χ^2. After the hit is added, the track parameters are transported to the next hit by the routine `klmn_trans`. After the track fit is done, the result is stored in the appropriate common block. If the fitter is run in a mode where the outward pass is activated, there will also be a pass where the fitter is run outward, and, if requested, hit residuals are calculated.

`klmn_trans` makes use of two routines to perform the transport. First, it uses the routine `t_to_t` for transport in vacuum, and gets the material affects from the routine `book_mater`. `book_mater` uses the routine `rad_lenght`, see Figure 9, to trace the track through the volumes. Since one of the most time consuming parts of track fit using the Kalman filter is finding which materials the track passes through, this information is saved from the first track fit so that it can be reused when the track is re-fitted. The routine `rad_length` uses a linearized recursive algorithm to search through the volumes. As discussed earlier the volumes are divided into two categories: the precision tracking volumes, and the chamber volumes. The precision tracking volumes are the volumes in the PTL or in the SVX, while the chamber volumes are the other volumes, typically the drift chamber volumes and the beam pipe. The precision tracking volumes are tabulated for faster particle tracing, and are handled by the routine `rad_lenght_svx`.

10.2 Program control

The fitter can perform a series of different fits. For instance, the fit can be run inwards (only) to obtain the optimal track parameters at the innermost point, or inwards and also outwards to also get the optimal set of track parameters at the outer most point. This also allows the possibility of calculating the hit residuals. There are also possibilities to turn off multiple scattering, energy loss, and straggling of the energy loss in the fit. This is mainly convenient for debugging. The different variables controlling the function of the code during a fit are summarized in Table 3. There is a brief discussion about the use of each variable.

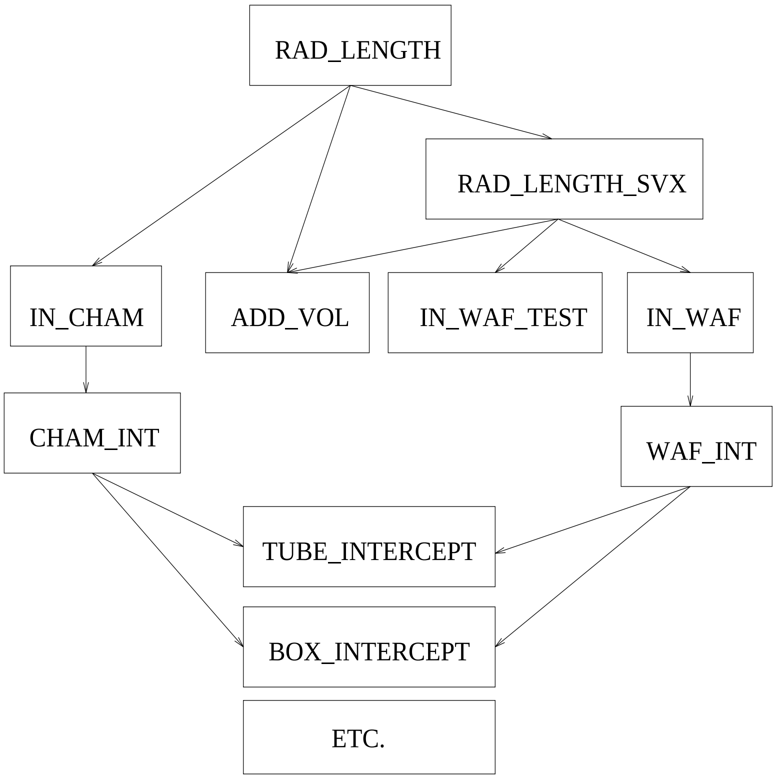

Figure 9: The calling sequence for tracing material. `rad_length` is the main routine for finding which volumes the track, approximated as a straight line segment, passed through.

Variable	Use
`klmnctrl_in`	This variable is either $+1$ or -1. It is set to $+1$ for an inward fit and -1 for an outward fit. Among other things, it controls if energy loss is added to the track during the inward pass, or if it is subtracted during the outward pass. It is also used to count up or down to find the next material segment to use when material is used.
`klmnctrl_buildmater`	If this variable is set to true the track is traced through the volumes which describe the geometry. Otherwise it is assumed that the track has been traced through the detector previously and the results are saved.
`klmnctrl_chisq`	If this variable is set to true, the χ^2 is calculated in the routine `klmn_add_hit`.
`klmnctrl_savepar`	If this variable is true, the track parameters are saved after each hit is added.
`klmnctrl_smooth`	When this variable is set to true, the fit is run outwards in addition to inward. This will give the track parameters at the outermost point, and will make it possible to calculate the hit residuals.
`klmnctrl_res`	Setting this variable to true enables the calculation of hit residuals.
`klmnctrl_reshit`	If this variable is set true, the calculated hit residual also includes the hit. Otherwise the residual for the hit is calculated based on the track parameters estimated from the other hits in the track fit.
`klmnctrl_part`	This is the particle hypothesis number tabulated in Table 4. The main use of this variable is to decide on how to calculate the physics properties of the particle's interaction with material in the detector.
`klmnctrl_mass`	Contains the mass of the particle for the hypothesis being fitted.
`klmnctrl_q`	Is either $+1$ or -1. The charge of the particle hypothesis being fitted.
`klmnctrl_noms`	If this variable is set to true the fitter will *not* account for multiple scattering.

`klmnctrl_noeloss`	If this variable is set to true the fitter will *not* account for any energy loss.
`klmnctrl_nostraggle`	If this variable is set to true the fitter will *not* account for straggling in the energy loss.
`start_type`	Either 0 or 1. If 0, the fit is started from scratch, and the variance matrix is initialized to be empty. If it is 1 the fit is continued and the variance matrix is initialized with the content of `v_start`, which must be supplied by the user. The seed parameters are taken as the track parameters with which to continue the fit.
`end_type`	Either 0 or 1. If 0, the track parameters are returned at the origin. Otherwise the track parameters are returned at the innermost hit.
`used_in_fit`	Flag to return which hits are used in the fit. Currently not implemented.
`imater_n`	Counts the number of line segments the track has been traced through when reusing the material list.
`nmater_n`	Counts the number of line segments the track has been traced through when building the material list.

Table 3: The table summarizes the variables that are used to control the fitter.

`klmnctrl_part`	Particle
1	e^- or e^+
2	μ
3	π
4	K
5	Pr

Table 4: The table describes the meaning of the variable `klmnctrl_part`. For `klmnctrl_part` = 1 the code decides which particle assignment to use based on the curvature of the seed track. For the other particles the physics properties are independent of the charge.

10.3 Main subroutines

To further help anyone who is trying to understand the functionality of the fitter we describe below how some of the main routines in the fitter work.

10.3.1 klmn_fit

A call to this routine is of the following form

$$\texttt{call klmn\_fit}(\texttt{part\_type}, \texttt{smooth}, \texttt{residuals}, \texttt{reshit}) \tag{113}$$

where part_type is an integer in the range 0 to 5. This specifies which particle type the fit will be performed for, see Table 4. If part_type = 0, the fit will be performed for all 5 particle hypothesis. The last three arguments are all of the type logical. If smooth is true the fit will also be run outwards. Hit residuals are calculated if residuals is set true, while if reshit is true the hit is included in the track fit before the hit residual is calculated. However, in order to do the hit deletion, currently a necessity in CLEO II, the smoothing is always performed and hit residuals are calculated. reshit is still used to decide whether or not to include the hit in the calculation of the hit residual.

ifit is a counter of how many times the track fit has been repeated due to the track being re-fitted after hits have been deleted. The variable is not used in any other way, then being passed to the hit deletion routine so it will know how many times it has been called for a given track. Before the track fit starts, there are protections against too large and small curvatures. For too large of a curvature, the fit is terminated and an error is returned. For too small curvature, the seed curvature is set to ± 0.01, and the fit is continued with this as the seed. This is to protect against numerical instabilities in tracks with a large radius of curvature. The error flags are cleared, and the number of deleted hits is set to zero. A routine called stop_init is called to initialize additional extra stops, to ensure that the treatment of the material is correct. The number of line segments, nmater_n, that the track has been traced through is set to zero. klnctrl_buildmater is set to true so that the track is traced through the volumes. A list of materials through which the track passes is build. This list will be used for the subsequent fits of this track. klmnctrl_chisq is set to true so that the χ^2 is calculated. klmnctrl_savepar, klmnctrl_smooth, and klmnctrl_res are set to true so that the track parameters at each hit are saved, and the fitter will do the outward pass, smoothing, and calculate the residuals. The residuals are calculated according to klmnctrl_reshit, which is initialized to the value of `reshit`. Before a fit is performed, the variables `klmnctrl_part`, `klmnctrl_q`, and `klmnctrl_mass` are initialized according to what particle type, `part_type` is being desired.

After this initialization is done, a track fit is performed by a call to the routine `cleofit_lin`. To prevent further fits from regenerating the material list, the variable `klmnctrl_buildmater` is set the false and the pointer `imater_n` is set to 1 so that the correct material will be accounted for in the next fit. Before the hit deletion routine is called, the counter `ifit` is increased. The hit deletion routine returns true if a hit was deleted, otherwise returning false. The fit is repeated until no more hits are deleted. If `part_type`=0, fits are performed for the other particle hypothesis.

10.3.2 `cleofit_lin`

This routine performs the actual track fit. It contains the main loop over the hits, and calls the necessary utility routines to perform the transport. First the variable `update` is set to false, so that the seed will not be updated until the track is well measured. The routine `klmn_init` is called to do the initialization of the track parameter. This includes setting the seed and initializing the error matrix. After this initialization, the actual track fit can be started. The first thing in the loop over the hits is to save the track parameters if the residuals are calculated without including the current hit. After this, the routine `klmn_add_hit` is called, in the format

$$\mathtt{callklmn\_add\_hit}(\mathtt{i}, \mathtt{eta0}, \mathtt{deta}, \mathtt{vetastar}, \mathtt{detaopt}, \mathtt{veta}) \tag{114}$$

where `i` is the number of the hit to add to the track. `eta0` is the seed track, `deta` is the linearized set of track parameters around `eta0`, and `vetastar` is the error matrix. These are the inputs to the fit. After the hit is added, the track is defined by `detaopt` and `veta`, still with respect to the same seed. After this the code checks to see if it should perform an update. If the track parameters where not saved before, they are saved now after we have added the hit to the track. After this is done, the track is transported to the next point. This transport might be broken into several different steps if additional stops was added. The routine that performs the actual transport is called `klmn_trans`, and is described below. After the transport is performed the routine proceeds to the next hit and repeats this until all hits are performed.

After the inward pass is completed the track parameters are saved. The track parameters are either saved at the point of closest approach to the origin, if `end_type` =0, or at the last hit, if `end_type`=1. If the fit is instructed to perform the outward pass, if `klmnctrl_smooth=true`, it proceeds to do the outward fit otherwise it returns to `klmn_fit`.

If an outward pass is requested, a call to `klmn_init_smooth` is performed to initialize the seed and the variance matrix. After this initialization, a fit is performed by looping, backwards, over the hits. The same steps as in the inward fit are performed, except that there is a call to the routine `klmn_smooth_d`, which performs the smoothing, i.e., the average between the inward and the outward pass of the fit. After all hits are added to the fit the track parameters are transported, in vacuum, to the origin, so that they are referenced with respect to the origin.

10.3.3 `klmn_add_hit`

This routine takes an new hit and uses it to estimate the new set of optimal parameters as well as the new variance matrix for the track which includes this hit. The first thing this routine does is to calculate the derivatives of a measured quantity, e.g., a drift distance with respect to the five track parameters. This derivative is evaluated with respect to the seed track, `eta0`. The derivatives are evaluated for the different types of hits by the routines `cleoaxialder`, `cleostereoder`, `cleocatder`, `cleochgdivder`, and `cleosvxder`. The routine which calculates the derivative also finds the projected measurement, `dpmin0`, for the seed track. Using the derivatives, the projected measurement, `dpmin`, for the track, `eta0`+`deta`, is calculated in the linear approximation.

After the derivatives and the projected measurement are found the new variance matrix, `veta`, and the new linearized track parameters, `detaopt`, are calculated, see Section 2.2 and 2.3.

If `klmnctrl_chisq` is set to true, the χ^2 is calculated, see Section 2.2. This requires an inversion of the variance matrix, performed by a call to the routine `invert_sym_d`. The χ^2, `chisq`, is updated and the routine returns to the calling function (`cleofit_lin`).

10.3.4 `klmn_trans`

This routine performs a transport of the track from one reference point to another, consisting of two steps. First, it transports the track parameters in vacuum, see Section 4.1, done by the call to the routine `t_to_t`. After this is done, the routine calls `book_mater` which returns the effect off material interactions, i.e., energy loss, multiple scattering, and straggling in energy loss.

The curvature of the track is corrected for the energy loss, however, to other track parameters are assumed to be unaffected by energy loss and MS. The error matrix is updated to account for the effects of multiple scattering, energy loss, and straggling in the energy loss, see Section 4.2 and 4.3.

10.3.5 `rad_length`

This routine finds the volumes which a straight line segment between two points passes through. The routine assumes that it knows which volume the first point is in, it returns the volume number the end point of the line segment is in. (There is a routine, `get_vol_num`, that finds the volume a given space point is in.) `rad_length` performs a "linearized" recursive algorithm to find the volumes a line goes through. First it searches the daughters of the volume that it is in, followed by a search of the daughters of the volumes it entered and so forth. After this search is exhausted, it goes back and checks if it exited the volume it started in. If so, it searches all the daughters of the mother volume of the volume the search started in, (and carefully avoids going back to the volume it came from). It also checks to see if it left this volume and so forth. Every time it enters a volume it accounts for the distance that it spends in this material. This is done with the routine `add_vol`. `rad_length` uses the routine `rad_lenght_svx` to trace trough the volumes in the precision tracking device. This routine works in a very similar fashion.

All distances calculated are normalized to the fraction of the length between the end points of the line segment. Before the program returns from this routine, the actual distances in the different materials are calculated by multiplying by `l0`, the length of the line segment. We also record the coordinates of the start and the end points of the line segment.

10.4 Initialization

The initialization of which volumes to be used is done in the routine `klmn_get_geom`. This initialization consists of reading in the volumes that will be used in the fitter, as well as reading in the material properties for the materials that make up these volumes.

It also has to initialize the magnetic field used in the fit. The magnetic field is, it self, actually never used by the fitter. Instead it needs the parameter `klmn_a` which allows the program to convert from curvature to $P_\perp$.

10.5 Miscellaneous

This section will discuss some other tips and tricks that we have learned.

One thing that we have had serious problems with has been that the compilers have had problems with the routine `t_to_t`. In particular, at several times we have had problems when the optimization of the compiler has been turned on. We have never been able to isolate the problem but compiling this routine without optimization has solved the problem. When this problem occurs, the error matrix transport is corrupted and we end up with an illegal error matrix. This causes, after the addition of several other hits to the fit, the program to crash. Most often it gives rise to a negative diagonal element in the variance matrix, causing a floating point invalid error. Tracing these errors down is made easier by having a routine, `check_var`, to test if the variance matrix has gone bad, see Appendix D.

11 Material interface

Earlier when we have, for example, considered the transport of track parameters in material, we have assumed that the expected energy loss, ΔE, the straggling, δE, and the multiple scattering, θ_0, were given. To calculate these, we use the information available about the track: its momentum, direction, position, and also the particle hypothesis. The direction and position of the particle are used to find which materials the particle passes through along its path. The momentum and the particle hypothesis, together with a model for the physics processes, allow the program to calculate the actual effects of the interactions of the particle with material. When a track is transported between two points, the material along the track segment is approximated as a straight line, and the routine `rad_length`, see Section 10.3.5, is used find which materials the track passes through. Given the length the particle travels in each material, the routine `dedx_line` calculates the effects of energy loss and multiple scattering.

There is a package of routines, `GEANE`, that uses the `GEANT` material description to transport error matrices and track parameters. There are two major reasons that we have not based our final fitter on this package. First it uses a different set of track parameters than the set used in this fitter. The track parameters that is used in this fitter are a natural choice for track fitting in a solenoidal field. The `GEANE` package was designed originally for a fixed target experiment and uses a slightly different set of track parameters. This would force a lot of changes of variables if the GEANE package was to be used, or a change to a slightly unnatural set of track parameters. Secondly, the GEANE package executes more slowly and would be harder to use. It has also been noticed that the GEANE package doesn't update the error matrix correctly for low momentum.

11.1 Energy loss calculation

We have adopted the method used in GEANT for the calculation of energy loss. The main motivations for using this calculation is that this calculation of the energy loss is accurate, and there is no need to integrate the energy loss as you must do if you use the simple Bethe-Block formula for low momentum particles. That we do not need to do an integration of the energy loss is an implicit fact in the way the calculation is done, see below. A second reason that we like to use this way of calculating the energy loss is that it allows us to precisely test the code as the data is generated exactly the same way as it is reconstructed. The disadvantage is that the calculation is quite time consuming.

For details about the calculation of the energy loss we refer the reader to the GEANT manual, as we will only briefly mention how the calculation works. The idea is that the stopping range of a particle is tabulated in logarithmic steps of kinetic energy. Given a particle energy, the stoping range is calculated in the material in question and and the distance in the material is subtracted from the stopping range. This new stoping range is inverted to give the new energy. The tables of stopping ranges are calculated at initialization of GEANT based on whatever physics processes the user tells the GEANT simulation to use. These tables are in the initialization of the fitter, see Section 9.1.2, copied into the fitters internal common block. So it is important that in the CLEVER job uses the right initialization.

11.2 Multiple scattering

Multiple scattering is also calculated in the same way as is done in GEANT. This calculation is very simple and has the nice feature of being "additive in squares", i.e., if you calculate the multiple scattering along a line or you split it into two pieces and combines the two segments contributions in square you get the same result. This calculation is based on a material parameter called χ_c which is tabulated for each material. Note that this calculation is not based on the radiation length.

11.3 Track tracing

This section describes the data structures used for storing the materials, also often called the volumes, which the detector is made from. This structure is similar to the CLEOG structure, but has some differences which will allow for faster execution. The main difference is that the precision tracking device, the PT or the SVD, has it's complicated volume structure tabulated in order to make the search of volumes faster.

A track is traced through the material as a series of straight lines between the added hits or the extra stop points which can be specified. These extra points are now coded directly into the routine `stop_init`. The main routine for tracing these line segments through the material banks is `rad_length`. This routine stores a list of distances through the materials which a track passes. Since the most time consuming part of the fit is to find the list of materials a track passes through, re-using this list for the subsequent fits of the same track saves time. In a standard 5 particle hypothesis fit, the list of materials are build for the pion hypothesis and then re-used for the other 4 particle hypothesis.

In Figure 9 the calling tree of the routine `rad_length` is shown. The two routines `rad_lenght` and `rad_lenght_svx` calls a series of low level routines that finds interceptions of lines with volumes. In the current CLEO geometry, either the PT-VD-DR or the SVX-VD-DR configuration, only the two volume types `TUBE` and `BOX` are used. But the following volume types are implemented: `TRD1`, `PGON`, `PARA`, and `PCON`. The routine `ADD_VOL` is used to which volumes the line segment is intercepting.

11.4 Material initialization

The volume structures are described by the variables in `chamber.inc` and in `wafer.inc`. These are initialized in the routine `klmn_get_geom` which is called in the beginning of each job. This routine also fills tables with the material properties needed when calculating the energy loss and multiple scattering of the particle. These calculations are done in the same way as in `CLEOG` as the calculations there are done in a state of the art manner. The energy loss calculation is based on a tabulation of the stopping distance, and to get a good result for large energy losses there is no need to do an integration over the Bethe-Block formula. The multiple scattering is not based on the radiation length, rather on a material parameter called χ_c and has the nice, for MC purposes, property that the result is independent of how the path is broken up in sub-sections. This is discussed in great detail in the `GEANT` manual. The set back to this is that the calculation has to be done in agreement with how it is done in `CLEOG`.

The tabulation of the precision tracking volumes are done at the beginning of each job by the routine `build_table`. The idea is to build a table of distances between the volume and a series of straight line segments coming out of the origin. The table is used by finding the line segment closest to the track and only searching the volumes that the track could have passed through based on the distance between the track and the line segment. For further details see the code: `rad_length_svx` and `init_table`.

11.5 Optimizing performance

The main transport routine `klmn_trans` performs a transport from a given point to another point in one step. All material interactions are accounted for as being at the final point in the transport step. The important thing in the transport of the track parameters is the correlation between scattering center and position. This means that for a longer transport an error is introduced in the treatment of the variance matrix. There are a few points in particular where this becomes a problem. When a transport is performed from one detector element to another and passes through a significant amount of material in the detector walls. If the next hit is far from the wall (hits are missing) there is a significant effect from getting the correlation between the scattering angle and the scattering center wrong. Also, when transporting from the innermost hit, through the beam pipe, to the interaction point a significant error occurs if all material is accounted for at the interaction point. For example, this means that there is no contribution to the error on the spatial point, there is an error only on the angle.

This problem is solved by adding a set of additional stoping points along the track to make the transport account more correctly for the material. These extra points are calculated in the routine `stop_init`. The points are specified as a list of different radii. Given the radii and seed, `stop_init` calculates the point which the transport will be performed to. These extra stopping points x and y coordinates are stored in `x_stop_mat` and `y_stop_mat`. If the variable `next_type(i)` is non zero, the transport from hit `i` to `i+1` will be broken into several steps. The first stopping point in the arrays `x_stop_mat` and `y_stop_mat` is given by `next_type(i)`, and the number of additional stopping points is given by `n_next_point(i)`. In the initialization routine `stop_init`, the variable `x_stop_mat` is filled with the radius of the stopping point and `next_type(i)` is negative. In the first track fit when the transport is done between hit `i` and `i+1`, the x and y coordinates are calculated and the `next_type` is made positive. This is done in `cleofit_lin`.

There are alternative ways of solving this. One that is used in other programs, (I do not know what `GEANE` is doing) is based on the idea of stopping at every volume boundary. This will take extra time and does not always solve the problem. A real solution would be to build a 2×2 matrix that correctly builds the correlations between the scattering center and the position, and then at the end of each transport step translate this into the change of the error matrix.

In order to optimize time performance, only material in the fiducial volume is there. This covers 85% of the volume. However for tracks very close to the beam-pipe all the material is not in the material list that is build by the fitter. One thing that can be done to remedy this is to use the `GEANE` routines for the small fraction of tracks that is affected by this.

12 Test of the Billoir fitter

Several test has been performed to show that the fitter is working. A few of these test will be discussed here. All discussion in this note will be based on "truth table" MC. This is so that we can decouple the problem of debugging the fitter with the (much harder) problem of having an aligned and understood tracking chamber. However, a large series of tests using the Kalman filter has been performed on real data and are written up in a CBX [4]. The interest here is simply to show that the fitter is working when the input is understood. This is demonstrated in Figure 10 where 2000 low momentum tracks (100 MeV/c) are fitted and the track parameter residuals and the probability of χ^2 distribution are shown to be in good agreement with the expectations.

We would also like to demonstrate here that the treatment of non-uniform magnetic fields is working. In Figure 11 the dashed line shows the results of track fits in which the magnetic field had been assumed to be uniform. The solid line is the result of fits that has corrected for the non-uniformity of the magnetic field.

The tests described here and lots of other things that we have considered over the course of the project makes us confident that the fitter is working. There is, however, always the question of whether the description of multiple scattering and energy loss is adequate. The multiple scattering is assumed to be Gaussian and the energy loss to have Gaussian straggling. It is well known that this is not the correct description of these physics processes. The multiple scattering has tails from from single scatterers and the energy loss has straggling that is more properly described by a Landau distribution. However, the Billoir fitter is doing an optimal job under these assumptions and the tails in these distributions do not degenerate the fitters performance to much. The largest reason for not understanding things like residual distributions in real data is more a problem of not understanding the detector, than the problem of non-gaussian tails in the multiple scattering or the straggling in the energy loss.

13 Conclusion

A final track fitter for CLEO II has been implemented using the Billoir technique. The fitter has been well tested on MC and shown to work well. The code has recently started to be used in the re-compress of the CLEO II dataset.

One of the technical problems that had to be overcome was to obtain stability in the start of the track fit. This was achieved using a linearization around a seed track. The track parameters determined from the pattern recognition are used as the seed for the Billoir fitter. This approach has shown to be stable against spurious hits at the start of the track fit as well as not leading to a bias of the estimated track parameters.

The track parameter transport requires knowledge about both the detector geometry and the material properties. This is used to calculate the effects of energy loss and multiple scattering on a particle when it traverses the detector. Both the geometry and the material properties are extracted from the GEANT simulation of the detector. This was done since it was considered that the GEANT description of the detector was the best available description. This information is processed and stored in common blocks internal to the fitter so that it can operate stand-alone from GEANT or CLEVER.

Since the CLEO pattern recognition is very generous to which hits are assigned to a track it is necessary to do hit deletion in the fitter. This is done after a full track fit has been performed and all the hit residuals are calculated. The same algorithm for eliminating hits are implemented in this fitter as was used in TF3FIT.

The magnetic field in CLEO is assumed to be uniform. However, this fitter has a mechanism allowing for non-uniform magnetic fields. This has been tried and showed to work well.

It is the hope that this fitter with its ability to correctly handle the effects of multiple scattering and energy loss will provide a better understanding of the error

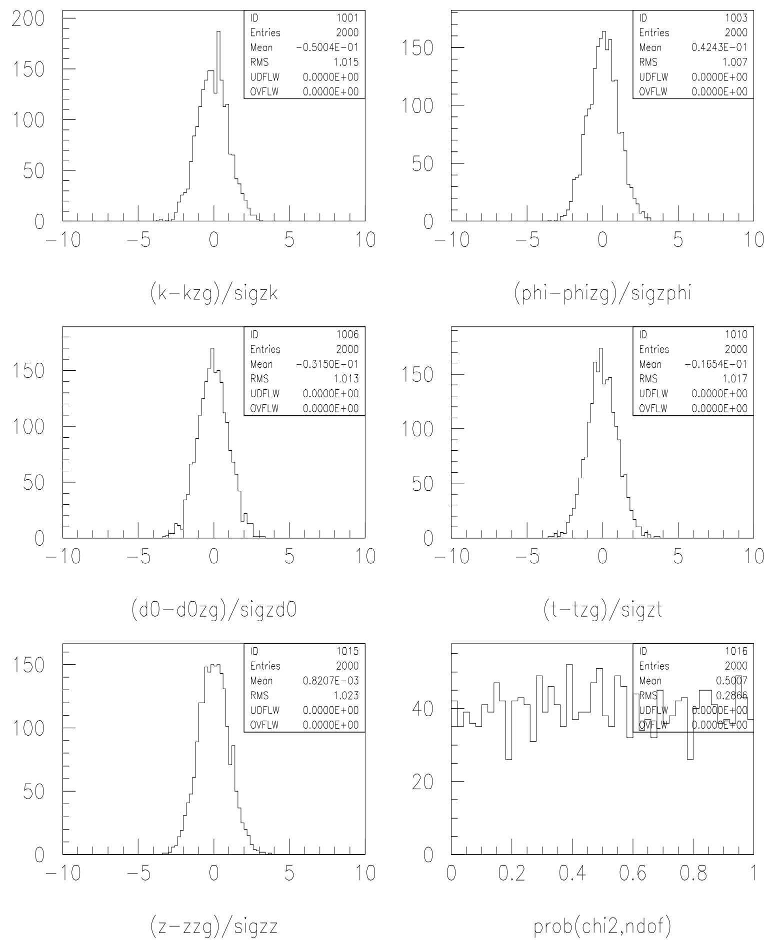

Figure 10: This plot shows the normalized track parameter residuals for a sample of 2000 100 MeV/c pions. The five track parameters are shown in the first five plots (K, ϕ, d_0, t, z) and the sixth plot is showing the probability of χ^2 distribution. All plots looks fine; the track parameter residuals are centered at zero and have a unit r.m.s. while the probability of χ^2 distribution is flat.

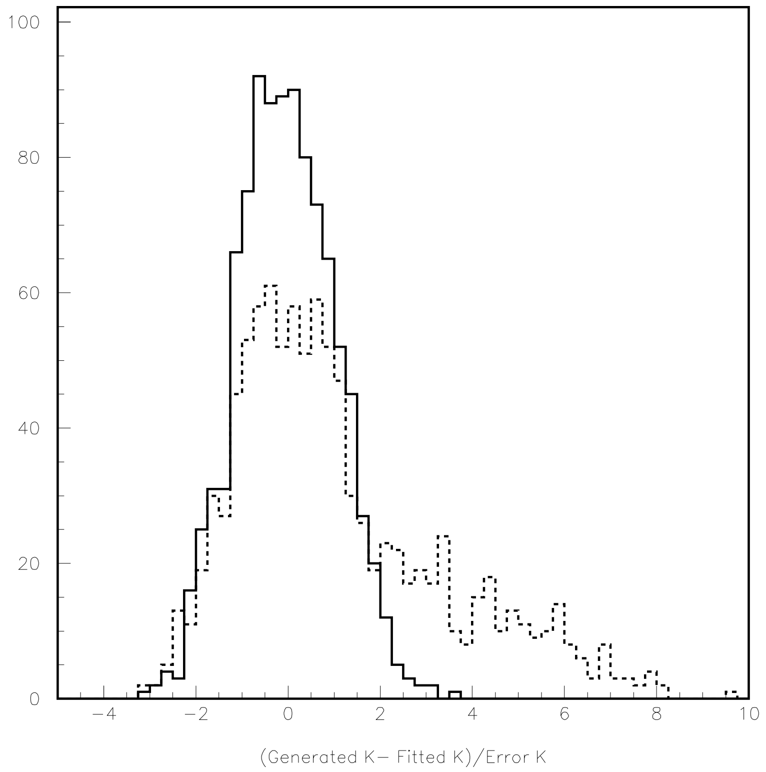

Figure 11: This solid line shows a fit in a non uniform magnetic field. The dashed line shows the same fit but with the fitter assuming the field to be uniform. (For help making this plot the authors would like to thank B. Lockman for the help of implementing this code in BaBar and for trying out the non-uniform field code.)

matrices. This is very important for understanding vertex constrained fits, and will certainly be even more so in the future with SVX data.

A Table of symbols

Symbol	Use
η	denotes track parameters $\eta = (K, \phi, d, t, z)$.
V_η	variance matrix for the track parameters η.
d	measurement, e.g., drift distance or a position on a silicon wafer. To emphasize that it is a measurement it is sometimes written d^{meas}. Projected measurements from track parameters are denoted by $d(\eta)$.
$\tilde{\eta}$	change in η from the addition of a measurement.
D	derivative of measurement with respect to track parameters, $D_i = \frac{\partial d}{\partial \eta_i}$.
σ	resolution on a measurement, i.e., $\sigma^2 = \langle (d - \bar{d})^2 \rangle$.
W	weight or information matrix, $W = V^{-1}$.
χ^2	the χ^2 of a fit.
η_0	seed track parameters for a track fit.
$\Delta\eta$	linearized track parameters, $\eta = \eta_0 + \Delta\eta$.
T	transport matrix $T = \frac{\partial \eta'}{\partial \eta}$.
q	charge of particle, $q = \pm 1$.
ρ	radius of curvature.
P	magnitude of momentum, P_x, P_y, and P_z are the cartesian components.
λ	the dip-angle, $t = \tan\lambda$. t is one of the track parameters.
c	is speed of light.
B	magnetic field in Tesla.
A	$cB/2$ relates curvature to momentum, $P_\perp = K/A$.
ψ	turning angle for helix parameterization.
$x_t,\ y_t,\ z_t$	positions on the track helix.
$x_c,\ y_c$	center of the track circle.
v	unit tangent vector of the track.
ΔE	energy loss.
δE	straggling in the energy loss.
m	mass of the particle.
$a,\ b$	derivatives in the update of the error matrix due to energy loss.
$x_{0_w},\ y_{0_w}$	coordinates of the stereo wires at the $z = 0$ plane.

$c_x,\ c_y$	components of the vector $v(c_x, c_y, 1)$ which gives the direction of the stereo wires.
ΔB	change in the magnetic field from the uniform field, $B_{\text{true}} = B\text{uniform} + \Delta B$.
ΔP	change in the momentum due to the residual magnetic field.
r	is the hit residuals.

B χ^2-fitting

This section will summarize the most important properties of χ^2-fitting. This is not meant to be a complete exposition of χ^2-fitting but it contains the things that is needed to develop the Billoir fit.

Before going in to the subject of this section a brief digression will be made to define some other important concepts like the variance matrix.

The variance matrix V_η (or simply V if not ambiguous) for a set of parameters η is defined as the expectation

$$V_{\eta_{ij}} \equiv \langle \delta\eta_i \delta\eta_j \rangle \ \equiv \langle (\hat{\eta}_i - \eta_i)(\hat{\eta}_j - \eta_j) \rangle \tag{115}$$

or in a more convenient matrix form

$$V_\eta \equiv \langle \delta\eta \delta\eta^t \rangle \ \equiv \langle (\hat{\eta} - \eta)(\hat{\eta} - \eta)^t \rangle, \tag{116}$$

where $\hat{\eta}$ is the mean, $\hat{\eta} = \langle \eta \rangle$.

Example.

Below we will use a variance matrix, V_y, which is the variance matrix for the measured points on a track. For N measured quantities $d_i, i = 1..N$ we have $V_{y_{ij}} = \langle \delta d_i \delta d_j \rangle$. We will now explicitly find the form of V_y for the configuration in Fig. 12. There is measurements of y at the positions x_1, x_2, and x_3 with precisions of σ_1, σ_2, and σ_3 respectively. This can for example be a simple model of a 3-layered silicon detector with the presence of M.S. in the wafers and in a beam pipe (at $x = x_0$). The rms of the M.S. is θ_i at x_i.

If all $\theta_i = 0$ then it is easy to see that

$$V_y = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}. \tag{117}$$

But what happens if there is multiple scattering? Let's look at an example,

$$V_{11} = \langle \delta y_1^{\text{meas}} \delta y_1^{\text{meas}} \rangle = \langle (\theta_0(x_1 - x_0) + \sigma_1)(\theta_0(x_1 - x_0) + \sigma_1) \rangle = \sigma_1^2 + \theta_0^2 (x_1 - x_0)^2. \tag{118}$$

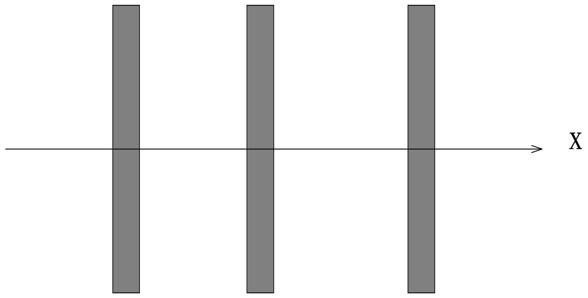

Figure 12: A simplified model of the silicon vertex detector, with three scattering layers.

In the presence of multiple scattering the off-diagonal elements do not vanish which is exemplified by the calculation of V_{12}

$$V_{12} = \langle \theta_0(x_1 - x_0)(\theta_0(x_2 - x_0) + \theta_1(x_2 - x_1)) \rangle = \theta_0^2(x_1 - x_0)(x_2 - x_0). \tag{119}$$

This leads to the following variance matrix

$$V_y = \begin{bmatrix} \sigma_1^2 + \theta_0^2(\Delta_{10})^2 & \theta_0^2(\Delta_{10})(\Delta_{20}) & \theta_0^2(\Delta_{10})(\Delta_{30}) \\ & \sigma_2^2 + \theta_0^2(\Delta_{10})^2 + \theta_1^2(\Delta_{12})^2 & \theta_0^2(\Delta_{20})(\Delta_{30}) + \theta_1^2(\Delta_{21})(\Delta_{31}) \\ \text{sym.} & & \sigma_3^2 + \theta_0^2(\Delta_{30})^2 + \theta_1^2(\Delta_{31})^2 + \theta_2^2(\Delta_{32})^2 \end{bmatrix} \tag{120}$$

where

$$\Delta_{ij} = x_i - x_j. \tag{121}$$

Another operation on variance matrices is what is often refered to as error propagation, but it is more correct to call it change of variables. Say that we have some set of parameters η' that is derived from another set η, with a variance matrix V_η, i.e. $\eta' = \eta'(\eta)$. We want to find $V_{\eta'}$. This is a straight forward calculation from the definition of the variance matrix:

$$V_{\eta'} = \langle \delta\eta' \delta\eta'^t \rangle = \langle (\frac{\partial \eta'}{\partial \eta}\delta\eta)(\frac{\partial \eta'}{\partial \eta}\delta\eta)^t \rangle = (\frac{\partial \eta'}{\partial \eta}) \langle \delta\eta\delta\eta^t \rangle (\frac{\partial \eta'}{\partial \eta})^t = T V_\eta T^t \tag{122}$$

where

$$T \equiv \frac{\partial \eta'}{\partial \eta}. \tag{123}$$

After establishing these basic properties of the variance matrix the discussion will now focus on the χ^2-fit. In the following we use the following notation, Roman indices $(i, j, k, ...)$ denotes indices running over measurements, Greek indices

$(\alpha, \beta, \gamma, ...)$ runs over the parameters. However the indices are often suppressed in favor of a more compact matrix notation.

The χ^2 is defined as the following quadratic form

$$\chi^2 \equiv [d^{\text{meas}} - d(\eta)]^t \, V_y^{-1} \, [d^{\text{meas}} - d(\eta)] \tag{124}$$

$$= [d_i^{\text{meas}} - d_i(\eta)] \left(V_y^{-1}\right)_{ij} \left[d_j^{\text{meas}} - d_j(\eta)\right]. \tag{125}$$

The parameters η are obtained by minimizing the χ^2 (as a function of η). This is in general a formidable task since $d(\eta)$ might be an arbitrarily complicated, non-linear function, of η. However a solution is straight forward in the case when $d(\eta)$ is a linear function, i.e., when

$$d_i(\eta) = d_i^0 + D_{i\alpha}\eta_\alpha. \tag{126}$$

This is good for most purposes since a general problem can always be linearized. This allows us to write the

$$\chi^2 = \left[\left(d_i^{\text{meas}} - d_i^0\right) - D_i\eta\right] \left(V_y^{-1}\right)_{ij} \left[\left(d_j^{\text{meas}} - d_j^0\right) - D_j\eta\right]. \tag{127}$$

We want to find the parameters η that minimizes this quadratic form, so we have to solve

$$0 = \frac{\partial\chi^2}{\partial\eta_\alpha} = -2D_{i\alpha} \left(V_y^{-1}\right)_{ij} \left[\left(d_j^{\text{meas}} - d_j^0\right) - D_j\eta\right]. \tag{128}$$

A little algebra shows that

$$\eta = \left(D^t V_y^{-1} D\right)^{-1} D^t V_y^{-1} \left(d^{\text{meas}} - d^0\right). \tag{129}$$

We also want to find what V_η which is found directly from its definition

$$V_\eta \equiv \langle \delta\eta\delta\eta^t \rangle \tag{130}$$

$$= \langle \left(D^t V_y^{-1} D\right)^{-1} D^t V_y^{-1} \delta d^{\text{meas}} \delta d^{\text{meas}t} V_y^{-1} D \left(D^t V_y^{-1} D\right)^{-1} \rangle \tag{131}$$

$$= \left(D^t V_y^{-1} D\right)^{-1} D^t V_y^{-1} \langle \delta d^{\text{meas}} \delta d^{\text{meas}t} \rangle V_y^{-1} D \left(D^t V_y^{-1} D\right)^{-1} \tag{132}$$

$$= \left(D^t V_y^{-1} D\right)^{-1} D^t V_y^{-1} V_y V_y^{-1} D \left(D^t V_y^{-1} D\right)^{-1} \tag{133}$$

$$= \left(D^t V_y^{-1} D\right)^{-1}. \tag{134}$$

This allows us to write

$$\eta = V_\eta D^t V_y^{-1} \left(d^{\text{meas}} - d^0\right) \tag{135}$$

and

$$\frac{\partial \eta}{\partial d^{\mathrm{meas}}} = V_\eta D^t V_y^{-1} \tag{136}$$

which is needed in the discussion of residuals.

C Transport derivatives

In this section the derivatives needed for the transport are derived. All derivatives are exact, no approximations are made that assumes that the transport distance is small compared to say the radius of curvature. As discussed in a Section 4.1 a transport is described by a change of the reference point of a track, this change is $(\Delta x, \Delta y, \Delta z)$. The initial parameters are described by $\eta = (\mathrm{K}, \phi, d, t, z)$ and the final parameters are denoted by $\eta' = (\mathrm{K}', \phi', d', t', z')$. The first task is to find $\eta' = \eta'(\eta)$. This is a geometrical problem, see Figure 4. With the shorthand notation

$$\begin{aligned}
q &= \frac{\mathrm{K}}{|\mathrm{K}|}, && (137)\\
x'_c &= \Delta x - \left(d + \frac{1}{2\mathrm{K}}\right) \sin\phi, && (138)\\
y'_c &= \Delta y + \left(d + \frac{1}{2\mathrm{K}}\right) \cos\phi, && (139)\\
r'_c &= \sqrt{x'^2_c + y'^2_c}, && (140)
\end{aligned}$$

we can now write

$$\begin{aligned}
\mathrm{K}' &= \mathrm{K}, && (141)\\
d' &= q r'_c - \frac{1}{2\mathrm{K}}, && (142)\\
\phi' &= \arctan\frac{x'_c}{y'_c}, && (143)\\
z' &= z + \Delta z - \frac{t\Delta\phi}{2\mathrm{K}}, && (144)\\
t' &= t, && (145)
\end{aligned}$$

where $\Delta\phi = \phi - \phi'$. It is now straight forward to evaluate the needed derivatives and the result is in the table below.

$\frac{\partial \rightarrow}{\partial \downarrow}$	K'	ϕ'	d'	t'	z'
K	1	$-\frac{a}{2\mathrm{K}^2 r_c'^2}$	$\left[1 - \frac{qb}{r_c'}\right] \frac{1}{2\mathrm{K}^2}$	0	$\frac{t}{2\mathrm{K}}\left(\frac{\Delta\phi}{\mathrm{K}} + \frac{\partial\phi'}{\partial\mathrm{K}}\right)$
ϕ	0	$\left(1+\frac{1}{2\mathrm{K}}\right)\frac{c}{r_c'^2}$	$-\left(1+\frac{1}{2\mathrm{K}}\right)\frac{qa}{r_c'^2}$	0	$\frac{t}{2\mathrm{K}}\left(\frac{\partial\phi'}{\partial\phi} - 1\right)$
d	0	$\frac{a}{r_c'^2}$	$\frac{qc}{r_c'}$	0	$\frac{t}{2\mathrm{K}}\frac{\partial\phi'}{\partial d}$
t	0	0	0	1	$-\frac{\Delta\phi}{2\mathrm{K}}$
z	0	0	0	0	1

In the table the following shorthands where used: $a = y_c' \sin\phi + x_c' \cos\phi$, $b = y_c' \cos\phi + x_c' \sin\phi$, and $c = y_c' \cos\phi - x_c' \sin\phi$.

D Test on goodness of variance matrices

During the development of the Kalman filter code it has on occasions been very convenient to have a test of whether a variance matrix is bad or not. With a bad variance matrix is meant a matrix that do not have the properties that is desired of a variance matrix. A variance matrix has a lot of properties of which some are obvious. First the diagonal elements has to be positive (can be allowed to be zero). Then the correlation coefficients have to have a norm less than or equal to 1. These are the obvious properties. But testing just these properties is not enough; there are further restrictions among the off diagonal elements. For example it is not so hard to convince oneself that in the case of three variables all three correlation coefficients can not be less than $-1/2$.

Given a variance matrix V, we would like to test if this matrix has the properties needed to be a variance matrix. This means that V can be written as an expectation value of a distribution

$$V = \langle \delta\eta \delta\eta^t \rangle. \tag{146}$$

The idea is now that there will be a linear transformation C such that $\delta\eta = Cz$ where z is a set of unit normal gaussian distributed variables which are uncorrelated. Further, C can be taken to be triangular.

This leads to $V = CC^t$. Therefore, a matrix that can be factorized into CC^t with C being triangular (and real) is a valid variance matrix.

A routine `check_var` is implemented to do this factorization and print out an error message if it is not possible to do. In particular it has been useful to apply this test to the variance matrix before and after it has been transported.

E Short summary of routines and commons

E.1 Subroutines

The routines in the Kalman filter are summarized with a short description of their functionality. For further documentation see the code.

`subroutine add_vol(ivol,s)`

This routine is used to add the distance `s` to the list of material that volume `ivol` is made of. If no previous volume has been added for this material a new entry is created. `nmater_n` is used to index which line segment the material is from. The routine is called from `rad_length` and `rad_length_svx`. Note that `s` can be less then zero and that this means that distance in a given material will be subtracted.

```
subroutine book_mater(eta_fr,x_fr,y_fr,z_fr, eta_to,x_to,y_to,z_to,
                      p,e,de,v2phi,v2t,v2k,vkt,ddk)
```

If `klmnctrl_buildmater` is true the routine finds the start and end points of the straight line segment in which it is finding the materials the track went through. It then uses the routine `rad_length` to find which material the line segment passed through. If `klmnctrl_buildmater` is false the routine just uses the volumes that a previous pass of the fit found. The routine `dedx_line` is used to find the energy loss and the multiple scattering that the track was subject to. Before returning from the routine some other quantities derived from the multiple scattering are calculated. Currently there is a simple calculation of the straggling there also. This calculation is probably to simple and possibly not even correct, it is likely to underestimate the effects of the straggling.

```
subroutine box_intercept(s_list,in_out_list,n_entries, dx,dy,dz,x1,
                         x2)
```

This routine finds the points at which a straight line enters and exits a box.

```
subroutine box_rmin_rmax(ivol,iwaf,rmin,rmax, phi_min,phi_max,zmin,
                         zmax)
```

This routine is called during initialization and calculates the maximum and minimum values of the radius, z position, and ϕ for a box.

`subroutine build_rmin_rmax`

This routine loops over all wafer elements and builds a table of the maximum and minimum values of the radius, z position, and ϕ for all elements.

`subroutine build_table`
Builds a table of the elements that are closest to a given ϕ angle. If the volume intercepts the ϕ slice the "distance" in radians is set to zero. The number of entries in the table, i.e., the number of ϕ slices is set by the variable `nlines`. The table only contains volumes that are at the lowest level in the precision tracking volume. The number of elements per ϕ slice controlled by the variable `ntable_waf`.

`subroutine calc_res(i,eta,veta)`
Calculates the residual of hit `i` using the track parameters `eta` and the error matrix `veta`. It also stores the error on the projected track in `sigma2_s(i)`.

`subroutine cham_int(icham,x1,x2,s_list,in_out_list,n_entries)`
Checks if the line from `x1` to `x2` goes through chamber `icham`. It builds a list of entry and exit points, `s_list` and `in_out_list`. The number of entry+ exit points are given in `n_entries`.

`subroutine check_var(v,txt)`
Checks if the variance matrix, `V`, is legal. It is not used in the code but is a very useful tool in debugging.

`subroutine cleoaxialder(k,d,phi,z,t,i,der,dw)`
Calculates the derivatives needed for adding an axial wire hit. Since it is assumed that the track has been transported to this point the form of the derivative is trivial, $\frac{\partial d}{\partial d} = 1$ and all others are zero.

`subroutine cleocatder(k,d,phi,z,t,i,der)`
Calculates the derivatives needed for adding a cathode hit. Since it is assumed that the track has been transported to this point the form of the derivative is trivial, $\frac{\partial d}{\partial z} = 1$ and all others are zero.

`subroutine cleofit_lin`
This is the main fit routine that performs one track fit. This routine is described in detail in Section 10.3.2.

`subroutine cleochgdivder(k,d0,phi,z,t,i,der)`
Calculates the derivatives needed for adding a charged division hit. Since it is assumed that the track has been transported to this point the form of the derivative is trivial, $\frac{\partial d}{\partial z} = 1$ and all others are zero.

`subroutine cleostereoder(k,d0,phi,z,t,i,der,dw)`
Calculates the derivatives needed for adding on a stereo wire hit. This routine first uses `stereo_dist` to find the point of closest approach of the track to the stereo wire. The calculation of the derivative is described in Section 5.2.

`subroutine cleosvxder(k,d0,phi,z,t,x0,v0,u0,derv,dpmin)`
Calculates the derivatives needed to add a hit in a plane wafer. This routine first performs an iteration to find the intercept of the track with the wafer-plane. Then it calculates the derivatives. The algorithm used is described in Section 5.5.

`subroutine cone_int(z1,r1,z2,r2,x1,v,s_list,n_inter)`
Finds the points where a straight line enters and exits a cone.

`subroutine cyl_int(k,d0,phi,z,t,r,x,y)`
Calculates the intercept, (`x`,`y`), of the track with a circle of radius `r`, centered at the origin.

`real function de(e,n_mater,step_s,beta)`
Calculates the energy loss of a particle with *kinetic* energy `e` as it goes through material `n_mater`. The length is given in `step_s` and is in *cm*. `beta` is the velocity of the particle and is used to check if a simple formula can be used. For `beta` < 0.8 the energy loss is calculated according to the full `GEANT` formulation.

`subroutine dedx_line(imater,p,e,delta_e,dphi2)`
Loops over the materials that was found in a line segment of the track and computes the energy loss and the multiple scattering expected from passing these materials. The energy loss is calculated by the routine `de`. `imater` is the line segment to evaluate the energy loss and the multiple scattering along. `p` and `e` is the momentum and energy of the particle. `delta_e` and `dphi2` are the expected energy loss and the square of the r.m.s. of the multiple scattering.

`logical function del_hit(n_call)`

This routine is called after a full fit is performed and the residuals are calculated. The purpose of this routine is to delete bad hits. If the track was stopped on the outward pass there are no hits deleted. The routine returns true if hits were deleted.

`function get_mater_file`
Reads in the material description file that can be written out by `klmn_get_geom`. This routine is used when transports are needed to be performed in DRIVER jobs.

`function get_run_num`
This routine returns the run number. It is useful when the duet common blocks cannot be include because of name conflicts between `GEANT` and `DUET`.

`function get_vol_num(x)`
This routine finds the volume that the point `x` is in.

```
subroutine in_cham(cham_num,x1,x2,s_list, cham_num_list,nentry,
                   n_entered,entered_list,s)
```
Checks if the line from `x1` to `x2` passes through the volume `cham_num`.`s` is the fraction of the distance between `x1` and `x2` that the line was inside the volume `cham_num`.

`function in_cham_vol(icham,x)`
Checks if the point `x` lies in volume `icham`. This routine is not general for the tube, i.e., it does not check ends of the tube.

`function in_pcon(ivol,x)`
Checks if the point `c` is in the volume `ivol` which has to be a PCON.

`subroutine in_quad(x,v,x1,x2,x3,x4,s_list,n_enter)`
checks if a straight line goes through the plane of a quad spanned in the four corners by `x1, x2, x3, x4`.

`function in_svx_vol(iwaf,x)`
Checks if point `x` is in wafer element `iwaf`. Not used.

```
subroutine in_waf(waf_num,x1,x2,s_list, waf_num_list,nentry,
                  n_entered,entered_list,s)
```
Checks if the line from `x1` to `x2` passes through the wafer `waf_num`. Returns same

information as `in_cham`.

`function in_waf_test(iwaf,rmin,rmax,phi_min,phi_max,zmin,zmax)`
Checks if line segment with the extreme values in the argument can intercept wafer element `iwaf`.

`subroutine init_table`
Initializes the table of distances to the precision tracking volume elements. It builds the list of ϕ values that the table is build around.

`subroutine invert_sym_d(a,d)`
Inverts a symmetric 5×5 matrix, `a` and returns the inverse as `d`.

`subroutine klmn_add_hit(i,eta0,deta,vetastar,detaopt,veta)`
This routine adds on hit, `i`, to the fit. It also builds the χ^2. This routine is described in Section 10.3.3.

`subroutine klmn_arc`
This routine calculates the arc length of a track. Not used.

`subroutine klmn_fit(part_type,smooth,residuals,reshit)` This is the main routine that is called to perform a fit. This routine is described in Section 10.3.1.

`subroutine klmn_get_geom`
Extracts the information from the `CLEOG` data structure.

`subroutine klmn_init(eta0,deta,vetastar)`
This routine do initialization needed for a inward fit. It initializes the seed and the error matrix.

`subroutine klmn_init_smooth(eta0,deta,vetastar)`
Performs the initialization needed for the outward fit, i.e., sets the seed and the initial error matrix.

`subroutine klmn_save_smooth_par(i,eta,veta)`
Saves the smoothed track parameters at hit `i`.

`subroutine klmn_save_track_par(i,eta0,detaopt,veta)`
Saves the track parameters on the inward pass at hit `i`.

`subroutine klmn_set_param(b)`
Set the B-field in the fitter, `klmn_bfield=b`, and calculates A, `klmn_a`.

`subroutine klmn_smooth_d(i,eta0,deta,vetastar,eta,veta)`
Performs the smoothing of the inward and outward track, i.e., averages the results from the two passes of the fitter.

```
subroutine klmn_trans(x_fr,y_fr,x_to,y_to,eta_fr,deta_fr, veta_fr,
                      eta_to,deta_to,veta_to)
```
Performs a transport of track parameters and error matrix. This routine is described in Section 10.3.4.

`subroutine klmn_xv(eta,psi,x,v)`
Calculates the position, `x`, and direction, `v`, of the track, `eta`, at the turning angle `psi`.

`subroutine line_rmin_rmax(x1,y1,vx,vy,rmin,rmax, phi_min,phi_max)`
Finds the extreme values of the radius, ϕ, and z for a straight line segment.

`subroutine para_intercept(s_list,in_out_list,n_inter, ivol,x,xto)`
Finds the intercept of a straight line with the para-volume.

```
subroutine para_rmin_rmax(ivol,iwaf,rmin,rmax, phi_min,phi_max,
                          zmin,zmax)
```
Finds the extreme values of radius, ϕ, and z for para-volume `ivol`.

`subroutine pcon_intercept(s_list,in_out_list,n_inter, ivol,x1,x2)`
Finds the intercept of a straight line with a poly cone.

`subroutine pgon_intercept(s_list,in_out_list,n_inter,ivol,x,xto)`
Find the intercept of a straight line with a polygon.

```
subroutine pgon_rmin_rmax(ivol,iwaf,rmin_w,rmax_w, phi_min,phi_max,
                          zmin, zmax)
```

Find extreme values of a polygon.

```
subroutine quad_rmin_rmax(x1,x2,x3,x4,rmin,rmax, phi_min,phi_max,
                          zmin,zmax)
```

Find the extreme values of a four cornered, plane, area spanned by `x1, x2, x3, x4`.

```
subroutine rad_length(x1,x2,cham_num_in,cham_num_out, l_list,s_list,
                      cham_num_list,nentry)
```

Traces a straight line from `x1` to `x2` and finds what volumes the particle goes through. This routine is described in Section 10.3.5.

```
subroutine rad_length_svx(x1,x2,waf_num_in,waf_num_out, l_list,
                          s_list,waf_num_list,nentry)
```

Traces a straight line through the wafer elements and finds the material a track passes through. This routine is called from `rad_length`. It is very similar in functionality to `rad_length` except that it also makes use of the information in the volume table to minimize the number of volumes it needs to search.

```
function stereo_dist(k,d0,phi,z,t,x0,y0,cx,cy,psi,v1xv2)
```

Finds the point of closest approach of a track to a stereo wire. It returns the signed distance. The algorithm used is described in Section 5.2.

```
subroutine stop_init
```

Calculates the extra points to stop at along a fit, e.g., for detector walls to make the transports correct.

```
subroutine swapr(r1,r2)
```

Swaps the value of `r1` and `r2`.

```
subroutine t_to_t(eta,deta,veta,etap, detap,vetap,dx,dy,dz,itrans)
```

Transports track parameters and error matrix through vacuum. The formulas used are discussed in Section 4.1 and Appendix C.

`subroutine t_to_t_c(eta,etap,dx,dy,dz,itrans)` Same as `t_to_t` except that it does not transport the error matrix.

`subroutine transform(i,xl,xg)` Transforms a local coordinate for wafer `i` to the global coordinate system.

`subroutine transform_to_local(rot,trans,xg,xl)`
Transform a global coordinate to the local coordinate system.

`subroutine transform_to_local_cham(i,xg,xl)`
Transform the global coordinate `xg` to the local coordinate `xl` for chamber `i`.

```
subroutine trd1[B_intercept(s_list,in_out_list,n_entries, dx1s,dx2s,
                            dys,dzs,x1s,x2s)
```
Finds the intercepts of a straight line from `x1s` to `x2s` with a TRD1 volume.

```
subroutine trd1_rmin_rmax(ivol,iwaf,rmin,rmax, phi_min,phi_max,
                          zmin,zmax)
```
Calculates the extreme values of a trd1 wafer element.

```
subroutine tube_intercept(s_list,in_out_list,n_inter, rmin,rmax,dz,
                          x1,x2)
```
Finds the entrance and exit points of a straight line from `x1` to `x2`.

```
subroutine tube_rmin_rmax(ivol,iwaf,rmin,rmax, phi_min,phi_max,
                          zmin,zmax)
```
Calculate the extreme values for the wafer `iwaf`.

```
subroutine user_trans(x_fr,y_fr,eta_fr,veta_fr, x_to,y_to,eta_to,
                      deta_to,veta_to,part_type)
```
This routine is intended as a user routine for transport of track parameters and error matrix for use for vertex constrained fits outside the beam pipe. The track parameters `eta_fr` and their variance matrix is transported from (`x_fr`,`y_fr`) to (`x_to`,`y_to`) and the track parameters are returned in `eta_to` with the error matrix `veta_to`. The transport is done for a particle specified by `part_type` ($1 = e$, $2 = \mu$,$3 = \pi$,$4 = K$,$5 = P$).

`subroutine waf_dist(iline,iwaf,dist)`
This routine is used to build the table of distances, in ϕ, between the different wafer elements, `iwaf`, and the lines, `iline`.

`subroutine waf_int(iwaf,x1,x2,s_list,in_out_list,n_entries)`
Checks if the straight line from `x1` to `x2` goes through wafer element `iwaf`.

`subroutine waf_rmin_rmax(iwaf,rmin,rmax,phi_min,phi_max,zmin,zmax)`
Finds the extreme values of wafer element `iwaf`.

E.2 Commons

A very brief summary of the various commons that are used in the fitter. This summary contains a broad description of the variables that are used. In the actual code all variables are described.

`chamber_n.inc`
This common block holds the information about the detector elements needed in the track fit, except for the elements that are part of the precision tracking device. It stores the position and orientation of each individual volume. However, the information about the geometry of the volumes are stored in `vol.inc`. Other information stored in this common is the position in the volume structure tree. This is copied from the GEANT volume structure.

`cleofitcom.inc`
This common contains the hit input to the fitter. Such as number of hits, type of hit, and a geometrical description of the hit.

`klmn_par.inc`
Common block to hold various parameters used in the fit. The magnetic field is also stored in this common block.

`klmncm.inc`
Contains the results of the fit at each of the hits along the track. If residuals have been calculated they are stored here.

`klmnctrl.inc`
This common block contains the variables that controls the fitter, e.g., which particle type is the fit being done for and what is calculated (χ^2, residuals, etc.).

`klmnerror.inc`

This common returns the error flags.

`klmnhitdel.inc`
Contains the parameters for hit deletion; how many hits can be deleted etc.

`klmntrk.inc`
The common is filled with the result from the fit. The final track parameters and the error matrices for the various track parameters are returned here. Also the χ^2 is stored here. This common also contains the seed track parameters that has to be filled before performing a track fit.

`mater_n.inc`
This common is used internally in the fitter to store which materials a track passed through so that it can be reused in sub sequential fits.

`mater_prop.inc`
This common stores the material properties for the volumes that are used in the fit. The information in this common is extracted from the GEANT commons at initialization of the fitter.

`mater_stop.inc`
This common block contains the information about where to stop the fit. This information is build in `klmn_init`. This is to make sure that too long steps are not taken when transporting the track parameters.

`table.inc`
The precision tracking volumes are tabulated in this common to allow for a faster access to the right volumes.

`vol.inc`
This common stores the information about the volumes, i.e., what type it is and what the geometry is.

`wafer_n.inc`
Contains the information about the position and orientation of the volumes in the precision tracking device. It also stores the information of the volume hierarchy.

References

[1] P. Billoir, Nucl. Instrum. Meth. **255**, 352 (1984).

[2] Phys. Rev. D 50, 1253 (1994).

[3] "How and Why Wonder Book of `DUET` Conventions", R. Kutschke, CSN 94-334.

[4] "Performance of the Kalman filter", R. Kutschke and A. Ryd, CBX 96-44.

[5] "Software Standards", Namjoshi, CSN 88/273.

[6] "Application of Filter Methods to the Reconstruction of Tracks and Vertices in Events of Experimental High Energy Physics", R. Fruhwirth, HEPHY-PUB-516/88, 1988 (Ph. D. Thesis).