Kalman-filter-based track fitting in non-uniform magnetic field with segment-wise helical track model

Bo Li\textsuperscript{a}, Keisuke Fujii\textsuperscript{b, \ast}, Yuanning Gao\textsuperscript{a}

\textsuperscript{a}Department of Engineering Physics, Tsinghua University, Beijing, 100084, China

\textsuperscript{b}High Energy Accelerator Research Organization (KEK), Tsukuba, 305-0801, Japan

Abstract

In the future International Linear Collider (ILC) experiment, high performance tracking is essential to its physics program including precision Higgs studies. One of major challenges for a detector such as the proposed International Large Detector (ILD) is to provide required momentum resolution in a magnetic field which is expected to be significantly non-uniform. The non-uniform magnetic field implies deviation from a helical track and hence requires the extension of a helical track model used for track fitting in a uniform magnetic field. In this paper, a segment-wise helical track model is introduced as such an extension. The segment-wise helical track model approximates the magnetic field between two nearby measurement sites to be uniform and steps between the two sites along a helix. The helix frame is then transformed according to the new magnetic field direction for the next step, so as to take into account the non-uniformity of the magnetic field. Details of the algorithm and mathematical aspects of the segment-wise helical track model in a Kalman-filter-based track fitting in the non-uniform magnetic field are elaborated. The new track model is implemented and successfully tested in the framework of the Kalman filter tracking software package, KalTest, which was originally developed for tracking in a uniform magnetic field.

Keywords:
track fitting, non-uniform magnetic field, Kalman filter, transformation

\ast Corresponding author.
Email address: keisuke.fujii@kek.jp (Keisuke Fujii)
1. Introduction

One of the primary goals of the next generation $e^+e^-$ collider, such as the International Linear Collider (ILC), is to make precise measurements of the properties of the Higgs boson thereby uncovering the secret of the electroweak symmetry breaking. This physics goal imposes great challenges on ILC detectors. For a main tracking detector, for instance, the momentum resolution $\delta(1/p_T)$ is required to be $O(10^{-4}) \,(\text{GeV}/c)^{-1}$ or better. The International Large Detector (ILD), one of the two conceptual detector designs currently pursued for the ILC experiments, uses a Time Projection Chamber (TPC) as its main tracking detector[1]. For the ILD TPC, the above required momentum resolution translates into about 200 sampling points along a track with a transverse spatial resolution of 100 $\mu$m or better over its full drift length of 2.2 m in a magnetic field of 3.5 T. New TPC readout techniques, based on Micro Pattern Gaseous Detector (MPGD) technologies having small $E \times B$ effect, good two-hit resolution, and excellent spatial resolution, provide a promising solution to satisfy the rigorous demand of the ILC.

In order to fulfill the required performance, however, the hardware R&Ds have to be backed by software developments that match the environment of the linear collider TPC. A Kalman filter software package, KalTest, has been successfully used for tracking in full detector simulations for physics feasibility studies as well as for tracking in test beam data taken with a Large Prototype (LP) TPC[2]. For the LP test, where the effect of the non-uniform magnetic field on the particle trajectory is small\(^1\), we could use a helical track model as used in the original KalTest. However, the future real LCTPC such as the ILD TPC must work in a non-uniform magnetic field with a non-uniformity up to a few percent[3]. In principle, Kalman filter algorithm itself is independent of the track model and can adapt to the non-uniform magnetic field. The most general solution is to implement a generic track model together with a Runge-Kutta track propagator. For the modest non-uniformity, this solution might not be optimal from the CPU time point of view. In this paper, another solution, a segment-wise helical

\(^1\) There are two major ways for the presence of a non-uniform magnetic field to manifest itself: (1) $E \times B$ effect that distorts the measured hit points and hence the apparent trajectory and (2) the real deviation of a track from a helical trajectory. The former effect can be corrected away in principle. As long as the latter is negligible, we can therefore use the helical track model.
track model, is proposed. It will be shown that the implementation of the  
segment-wise helical track model that replaces the original simple helical  
track model allows us to successfully realize Kalman-filter-based track fitting  
in a moderate non-uniform magnetic field with the minimal change to the  
original KalTest package. The time consumption of track fitting in the non-  
uniform magnetic field will also be discussed.

2. KalTest

KalTest is a ROOT\cite{4} based Kalman filter software package written in  
C++ for track fitting in high energy physics experiments. Comparing with  
the least square fitting method, the Kalman filter has great advantages in  
track fitting\cite{5}. The basic formulae and their implementation in KalTest are  
summarized in this section, while details can be found in KalTest manual\cite{6}.

2.1. Kalman filter

The Kalman filter handles a system that evolves according to an equation  
of motion (system equation) under the influence of random disturbance (pro-  
cess noise). It is designed to provide the optimal estimate of the system’s  
state at a given point from the information collected at multiple observation  
points (measurement sites). Suppose that there are \( n \) measurement sites  
(\( k = 1, \ldots, n \)) and the state of the system at site (\( k \)) can be specified by a  
p- dimensional column vector (state vector) \( \bar{a}_k \), where the bar indicates that  
it is the true state vector without any measurement error. The system equa-  
tion that describes the evolution of the state at site (\( k - 1 \)) to the next one,  
site (\( k \)), can be written in the form:

\[
\bar{a}_k = f_{k-1}(\bar{a}_{k-1}) + w_{k-1}, \tag{1}
\]

where \( f_{k-1}(\bar{a}_{k-1}) \) is a state propagator which expresses a smooth and deter-  
minstic motion that would take place if there were no process noise, and  
\( w_{k-1} \) is the process noise term due to the random disturbance. It is assumed  
that the process noise is unbiased and has a covariance given by

\[
Q_{k-1} \equiv < w_{k-1} w_{k-1}^T > . \tag{2}
\]

At each site, we measure some observables about the system. The values  
of these observables comprise a \( m \)-dimensional column vector (measurement
vector) \( \mathbf{m}_k \). Its relation to the state vector \( \bar{\mathbf{a}}_k \) at site \( (k) \) is called a measurement equation:

\[
\mathbf{m}_k = h_k(\bar{\mathbf{a}}_k) + \epsilon_k,
\]

in which \( h_k(\bar{\mathbf{a}}_k) \) is a projector which gives, as a function of the state vector, the measurement vector you would expect for an ideal measurement with no measurement error, and \( \epsilon_k \) is the random measurement error (measurement noise) unavoidable in practice. We assume here that systematic errors such as those from misalignment of detectors have been corrected and hence the random measurement noise is unbiased and having a covariance given by

\[
V_k \equiv (G_k)^{-1} \equiv \langle \epsilon_k \epsilon_k^T \rangle.
\]

In the Kalman filter process, two operations, prediction and filtering, are needed at each site to proceed. The state vector prediction is the extrapolation of \( \mathbf{a}_{k-1} \) to the next site by using Eq.(1):

\[
\mathbf{a}_k = f_{k-1}(\mathbf{a}_{k-1}) \equiv f_{k-1}(\mathbf{a}_{k-1}),
\]

where the superscripts \( (k-1) \) to the state vectors indicate that the state vectors are estimated using the information up to site \( (k-1) \). In what follows we will omit the superscript, if the superscript coincides with the subscript as \( \mathbf{a}_k \equiv \mathbf{a}_{k-1} \).

The covariance matrix for \( \mathbf{a}_{k-1} \) is defined by

\[
\mathbf{C}_{k-1} \equiv \left\langle (\mathbf{a}_{k-1} - \bar{\mathbf{a}}_{k-1})(\mathbf{a}_{k-1} - \bar{\mathbf{a}}_{k-1})^T \right\rangle,
\]

then the prediction for the covariance matrix at site \( k \) is given by

\[
\mathbf{C}_k \equiv F_{k-1}\mathbf{C}_{k-1}F_{k-1}^T + Q_{k-1},
\]

where

\[
F_{k-1} \equiv \frac{\partial f_{k-1}}{\partial \mathbf{a}_{k-1}}
\]

is called a propagation matrix.

In the filtering step, the predicted state vector at site \( (k) \) is updated by taking into account the pull that is defined to be the difference between the measured and the predicted measurement vectors, \( \mathbf{m}_k - h_k(\mathbf{a}_{k-1}^{(k-1)}) \), as

\[
\mathbf{a}_k = \mathbf{a}_{k-1}^{(k-1)} + K_k \left( \mathbf{m}_k - h_k(\mathbf{a}_{k-1}^{(k-1)}) \right),
\]
in which, \( K_k \) is the gain matrix given by

\[
K_k = \left[ (C_k^{k-1})^{-1} + H_k^T G_k H_k \right]^{-1} H_k^T G_k
\]  

(10)

with \( H_k \) defined by

\[
H_k \equiv \frac{\partial h_k}{\partial a_{k-1}^k},
\]  

(11)

which is called the measurement matrix.

After all the \( n \) sites are filtered, the state vector at site \( k (k < n) \) can be reevaluated by including the information at subsequent sites: \( k + 1 \) to \( n \). This process is called Smoothing. The smoothed state at site \( k \) is obtained by the following backward recurrence formula

\[
\begin{cases}
    a_n^k = a_k + A_k (a_{k+1}^n - a_{k+1}^k) \\
    A_k = C_k F_k^T (C_{k+1}^k)^{-1}
\end{cases}
\]  

(12)

which gives the smoothed state at site \( k \) in terms of the smoothed state at site \( k + 1 \), the predicted state at site \( k + 1 \), and the filtered state at site \( k \).

2.2. Helical track parametrization

In a uniform magnetic field a charged particle follows a helical trajectory. If we set our coordinate system in such a way that the magnetic field points to the \( z \) axis direction, the helix can be parametrized as

\[
\begin{cases}
    x = x_0 + d_\rho \cos \phi_0 + \frac{\alpha}{\kappa} (\cos \phi_0 - \cos (\phi_0 + \phi)) \\
    y = y_0 + d_\rho \sin \phi_0 + \frac{\alpha}{\kappa} (\sin \phi_0 - \sin (\phi_0 + \phi)) \\
    z = z_0 + d_z - \frac{\alpha}{\kappa} \tan \lambda \cdot \phi
\end{cases}
\]  

(13)

where \( \mathbf{x}_0 = (x_0, y_0, z_0) \) is an arbitrary reference point on which three \( (d_\rho, \phi_0, d_z) \) out of the five helix parameters, \( d_\rho, \phi_0, \kappa, d_z, \) and \( \tan \lambda \), depend and \( \alpha \) is a constant defined by \( \alpha \equiv 1/cB \) with \( B \) and \( c \) being the magnetic field and the speed of light, respectively. Since the reference point \( \mathbf{x}_0 \) is arbitrary, we can take it to be the measured hit point at each site, say, site \( (k) \) and call it a pivot. Then \( \phi \) measures the deflection angle from the pivot. The geometrical meanings of the five helix parameters are depicted in Fig.1. Notice that \( \rho \equiv \alpha/\kappa \) is the radius of the helix singed by the particle charge, while \( \kappa \equiv Q/p_t \) with \( Q \) being the charge in units of the elementary charge.
Figure 1: Helical track parametrization in KalTest. \( \mathbf{x} = (x, y, z) \) is a point on the helix, \( \mathbf{x}_0 = (x_0, y_0, z_0) \) a reference point usually taken to be a hit point. \( d_\rho \) is the distance of the helix from the pivot in the \( x-y \) plane, \( \phi_0 \) the azimuthal angle of the pivot with respect to the center of the helix, \( d_z \) the distance of the helix from the pivot in the \( z \) direction, and \( \lambda \) stands for the dip angle.

and \( p_t \) being the transverse momentum. The five parameters in Eq. (13) are combined to make a concrete state vector

\[
\mathbf{a}_k = \left( d_\rho, \phi_0, \kappa, d_z, \tan \lambda \right)^T.
\]

The reason to use \( \kappa = Q/p_t \) instead of \( \rho \) or \( p_t \) as a helix parameter is to allow continuous change of the state vector for a high momentum track for which the curvature might change its sign during the track fitting. Notice that there is a difference of \( \pi \) in the definition of \( \phi_0 \), depending on the track charge. This is to avoid discontinuity of \( \phi_0 \) that would happen if \( \kappa \) changes its sign during track fitting.

2.3. Implementation

Notice that the formalism presented above is generic, though the concrete forms of the state propagator \( f_{k-1}(\mathbf{a}_{k-1}) \), the projector \( h_k(\mathbf{a}_{k-1}) \), and their derivatives, \( F_{k-1} \) and \( H_k \), depend on the track model and the geometry and distribution of the measurement sites. In order to keep the application scope of this generic formalism as wide as possible, KalTest is designed to extract this generic part as a separate sub-package (KalLib) that forms an abstract layer consisting of abstract base classes that implement the generic Kalman filter algorithm. In order to realize Kalman-filter-based track fitting, we then inherit from the generic base classes in the abstract base class library KalLib.
and implement their pure virtual methods for track fitting in a tracking de-
tector consisting of measurement layers. The resultant Kalman-filter-based
track fitter library is KalTrackLib. KalTrackLib, however, should not de-
pend on any particular track model or shape or coordinate system of any
measurement layer so as to accommodate variety of tracking devices which
may coexist in the same tracking system. KalTrackLib is hence designed
to interact with concrete track model or concrete measurement layer classes
always through interface classes that represent an abstract track or an ab-
stract measurement layer class. The implementation of concrete track models
(helix, straight line, etc.) and measurement surfaces (cylinder, hyperbolic,
flat plane, conical surface, etc.) are separated out into a geometry library
GeomLib.

Figure 2 shows some major classes contained in these three packages. In
the KalLib library, TVKalSystem is implemented as an array of TVKalSite-
derived objects and represents a collection of information gathered at each
measurement site. Each TVKalSite-derived object can contain up to three
Kalman filter states (i.e. predicted, filtered, and smoothed), each of them
being an instance of some concrete class inheriting from TVKalState. The
functions TVKalState::Propagate() and TVKalSite::Filter() implement
Eqs.(7) and (9), respectively. The pure virtual functions such as TVKalSt-
ate::MoveTo(), TVKalSite::CalcExpectedMeasVec(), and TVKalSite::-
CalcMeasVecDerivative() declared in the two classes are implemented in
the corresponding derived classes of KalTrackLib. These concrete functions
in the derived classes interact with a track model or a measurement layer
through the abstract layer made of TVTrack, TVSurface, and TVMeasLayer
and are still quite generic. The concrete geometrical features of the track or
the measurement layers are supplied by concrete classes in GeomLib, where,
for instance, the helical track model defined by Eq.(13) is implemented in a
class called THelicalTrack derived from TVTrack, which actually propagates
and calculates the concrete value of the state vector \( \mathbf{a} \), the propagation ma-
trix \( F \), and the covariance matrix \( C \) and a cylindrical surface is implemented
in a class called TCylinder inheriting from TVSurface.

The architectural design of KalTest thus minimizes the number of user-
implemented classes to the following three:

- MeasLayer: a measurement layer object that multiply inherits from a
  concrete shape class such as TCylinder and the abstract measurement
  layer class TVMeasLayer.
- **KalDetector**: a class derived from TVKalDetector, which is implemented as an array of TVMeasLayer pointers and holds MeasLayers with any shape and coordinate system as well as materials.

- **Hit**: a coordinate vector class as defined by the MeasLayer class, which inherits from TVTrackHit.

Notice that, by design, KalTest allows site-to-site change of track models.

Figure 2: Class diagram of KalTest. KalLib contains the generic Kalman filter procedure. The pure virtual functions specific to tracking are implemented in KalTrackLib. Basic geometrical objects to express detector configuration or to represent track models are implemented in GeomLib.

Since the original helical track model in KalTest is valid only in a uniform magnetic field, the result of track fitting will not be satisfactory if the track
is generated in a significantly non-uniform magnetic field, as we will see in Sec. 4. A new class in the KalTrackLib library, TTrackFrame, which contains the algorithm for transforming a coordinate frame in a non-uniform magnetic field, can solve this issue and will be described in the next section.

3. Algorithm for non-uniform magnetic field

3.1. Basic idea

If the non-uniformity of the magnetic field is not too large, we can assume that the magnetic field between two measurement layers is approximately uniform. We can then propagate our track from one measurement layer to the next using the helix model we discussed above. When the track reaches the next layer, we update the magnetic field in order to take the non-uniformity into account, and propagate the track with the updated magnetic field. The track produced this way is hence segment-wise helical and hereafter called a segment-wise helical track. Notice that the direction of the magnetic field also changes as well as its magnitude in general. We therefore need to attach, to each track segment, a local frame having its $z$ axis pointing to the magnetic field direction so as to use our helix parametrization defined in Eq.(13). At the end of each step, we hence update the frame to make its $z$ axis parallel with the magnetic field there and transform the propagated state vector to this new updated frame.

This new track propagation procedure is illustrated in Fig. 3. Firstly, the state vector $\mathbf{a} = \mathbf{a}_k$ at layer $(k)$ is propagated to $\mathbf{a'}$ at layer $(k+1)$ along helix $(k)$ parametrized by Eq.(13). The magnetic field at the intersection of the helix and layer $(k+1)$ is then calculated. With the new magnetic

![Figure 3: Basic idea of transforming the state vector and the associated frame between two nearby layers.](image-url)
field and the one at the last intersection, the matrix for rotating the frame
can be obtained. Since the state vector is not easy to transform directly, it
is converted to the momentum vector. Applying the rotation matrix to the
momentum vector and then using the rotated momentum, we can calculate
the predicted state vector \( a'' \) in the new frame. Notice that this defines a new
modified state propagator as \( a'' \equiv a_{k+1}^k = f_{k}^{\text{mod}}(a_k) \). Finally, we update the
predicted state vector in the new frame with Eq.(9), getting the filtered state
vector \( a''' \equiv a_{k+1} \). The same procedure is repeated for subsequent steps.

Since the propagation procedure or equivalently the propagator function
is modified, the propagator matrix should also be modified accordingly. We
will explain how in the next subsection.

3.2. Transformation

The transformation between the two frames consists of a shift and a
rotation as is illustrated in Fig.4. The shift is given by the vector \( \Delta d_k \) and
parallel-transports the \( xyz \)-frame at the starting point on layer \((k)\) to the
new \( x'y'z' \)-frame at the predicted intersection on layer \((k + 1)\). The \( x'y'z' \)-
frame is then rotated so as to make the \( z'' \) axis point to the new magnetic
field. This rotation \( \Delta R \) is defined as follows:

\[
\Delta R = \Delta R_{z'}(-\phi) \Delta R_{y'}(\theta) \Delta R_{z'}(\phi),
\]

where the \( x'y'z' \)-frame is first rotated around the \( z' \) axis by the angle \( \phi \) to
bring the \( y' \) axis to a tentative \( y'' \) axis so that it becomes perpendicular to
the plane spanned by the \( z' \) axis and the new magnetic field direction; the
frame is then rotated by the angle \( \theta \) around the tentative \( y'' \) axis to make the
\( z'' \) axis point to the new magnetic field direction; finally the frame is rotated
again this time around the \( z'' \) axis by the angle \(-\phi \) to bring the tentative \( y'' \)
axis to its final direction.

Since the rotation is passive for a vector, the rotation matrices are given
by

\[
\Delta R_{z'}(\phi) = \begin{pmatrix}
\cos \phi & \sin \phi & 0 \\
-\sin \phi & \cos \phi & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

and

\[
\Delta R_{y'}(\theta) = \begin{pmatrix}
\cos \theta & 0 & -\sin \theta \\
0 & 1 & 0 \\
\sin \theta & 0 & \cos \theta
\end{pmatrix}.
\]
Figure 4: Transformation from one frame to the next. The $\theta$ and $\phi$ angles are determined by the magnetic field directions at the position $O_k$ and $O_{k+1}$.

Considering the shift and the rotation, a local position vector $x_k$ in frame $(k)$ can be transformed to a local position vector in frame $(k+1)$ by

$$x_{k+1} = \Delta R_k (x_k - \Delta d_k). \quad (15)$$

We also need a transformation of a vector from a global frame to the local frame:

$$x_{k+1} = R_k (\tilde{x} - d_k), \quad (16)$$

in which $\tilde{x}$ is the corresponding vector defined in the global frame. In Eq.(15), $x_k$ is defined in local frame $(k)$, and it can be transformed from the global vector by

$$x_k = R_{k-1} (\tilde{x} - d_{k-1}) \quad (17)$$

Substituting Eq.(17) into Eq.(15)

$$x_{k+1} = \Delta R_k [R_{k-1} (\tilde{x} - d_{k-1}) - \Delta d_k],$$

then the global rotation matrix $R_k$ and the shift vector $d_k$ can be represented
by the following recurrence formulae:
\[
\begin{align*}
R_k &= \Delta R_k R_{k-1} \\
d_k &= d_{k-1} + R_{k-1}^{-1} \Delta d_k
\end{align*}
\]  
(18)

The local magnetic field in frame \((k)\) is transformed simply by
\[
B(x) = R_k \tilde{B}(\tilde{x}),
\]  
(19)
in which \(\tilde{B}(\tilde{x})\) is the magnetic field in the global frame.

3.3. Propagator matrix

The total propagation described in Section 3.1 can be represented mathematically by
\[
\begin{align*}
\alpha' &= f_k(\alpha_k) \\
p &= c(\alpha') \\
p' &= t(p) \\
\alpha'' &= c^{-1}(p')
\end{align*}
\]  
(20)

where, the function \(f_k\) is the original state vector propagation function for a uniform magnetic field, \(c\) is a function which converts a state vector to the corresponding momentum with \(c^{-1}\) being its inverse, and the function \(t\) is a rotation given by the rotation matrix \(\Delta R\).

The momentum can be calculated using Eq.(13):
\[
p = - \left( \frac{Q}{\alpha} \right) \frac{dx}{d\phi} = \frac{1}{|\kappa|} \begin{pmatrix} -\sin \phi_0 \\ \cos \phi_0 \\ \tan \lambda \end{pmatrix}.
\]  
(21)

As mentioned before, the sign of \(\kappa\) is the sign of the particle charge. The momentum vectors in the two successive local frames are related by \(\Delta R\):
\[
p' = \Delta R p.
\]  
(22)

During the transformation, the intersection (taken to be the pivot) is on the helix. Therefore the two distance components, \(d_\rho\) and \(d_z\), are zero, while the other three components, \(\phi_0\), \(\kappa\), and \(\tan \lambda\), can easily be solved from the momentum by their definitions:
\[
\begin{align*}
\alpha'' &= \begin{pmatrix} d_\rho \\ \frac{\tan^{-1}(-p'_x, p'_y)}{s_\kappa} \\ d_z \\ \frac{p'_z}{(p'^2_x + p'^2_y)^{\frac{1}{2}}} \end{pmatrix},
\end{align*}
\]  
(23)
where the sign of \( \kappa \) in the last local frame (i.e. \( s_\kappa \equiv \text{sgn}(\kappa) \)) is used, since it is safe to assume that the magnetic field direction will not be reversed as long as the magnetic field varies moderately.

According to Eq.(20), the modified propagator matrix is given by

\[
F_{k}^{\text{mod}} \equiv \frac{\partial f_{k}^{\text{mod}}(a)}{\partial a} = \frac{\partial a''}{\partial a}, \tag{24}
\]

which is calculated to be

\[
F_{k}^{\text{mod}} = \frac{\partial a''}{\partial p'} \frac{\partial p'}{\partial p} \frac{\partial p'}{\partial a'} \frac{\partial a'}{\partial a} = F_{k}^{\text{rot}} F_k. \tag{25}
\]

The original propagator matrix \( F_k \) is known in the original KalTest already. The other three matrices in \( F_{k}^{\text{rot}} \) are given in Appendix A.

The transformation algorithm described in this section is implemented in the class TTrackFrame. In order to promote our THelicalTrack class to accommodate the segment-wise helical track model for non-uniform magnetic field, a TTrackFrame object is added as its data member. With the help of TTrackFrame, the THelicalTrack can calculate the global position, the modified state vector, and the modified propagator matrix. This way, all the modifications for the segment-wise helical track implementation are hidden in THelicalTrack and are invisible to the other classes. Because of the locality of the segment-wise helical track model implementation, it can be seen that the track smoothing is still valid from Eq.(12). Further it should be relatively easily ported into any Kalman filter package other than KalTest.

4. Test of algorithm

4.1. Simulation conditions

After the algorithm for track fitting in a non-uniform magnetic field was implemented based on the original KalTest code, its track reconstruction performance has been tested intensively. Firstly, the mean and the sigma of momentum distributions were compared for the original code and the new code. As expected, the results are identical if tracks are generated and reconstructed in a uniform magnetic field.

For the purpose of testing track reconstruction in a non-uniform magnetic field, we assumed the following (artificially non-uniform) magnetic field \( \mathbf{B} \):

\[
\begin{align*}
B_x &= B_0 kxz \\
B_y &= B_0 kyz \\
B_z &= B_0 (1 - k z^2)
\end{align*}
\]
in which, $k = \frac{k_0}{r_{\text{max}}}$, $B_0 = 3$ T, $z_{\text{max}} = r_{\text{max}} = 3000$ mm. The coefficient $k$ can be set to several values for different non-uniformities. In simulation, we assumed a tracker having 251 measurement layers with a layer-to-layer distance of 6 mm, an inner radius of 300 mm, and outer radius of 1800 mm. The detector geometry is similar to the configuration of the LC TPC in the ILD detector concept[1].

To simulate tracks in the non-uniform magnetic field, a Runge-Kutta propagator class in ROOT[4], `TEveTrackPropagator`, was used. In this class, the position is a function of the length along the track and the crossing point of the track with a measurement layer is calculated by the bisection method. The crossing point was smears to make a hit position according to the detector spatial resolution (100 µm) in the $r\phi$ plane. For the track generation, we generated dip angles and an azimuthal angles uniformly in $\lambda \in [0, 0.5]$ and $\phi \in [0, 2\pi]$, respectively. In Fig.5, a typical event display is shown for a track generated in the non-uniform magnetic field with $k_0 = 5$.

In the same figure, for comparison, a track with the same initial momentum but in a uniform magnetic field ($k_0 = 0$) is also shown. Notice that the third component of $\mathbf{B}$ depends on the $z$ position quadratically in our field assumption, therefore the discrepancy of the two tracks increases as their $z$ position becomes large.

![Figure 5: Event display. Tracks with 2 GeV/c simulated in a non-uniform magnetic field (red curve: $k_0 = 5$) and uniform magnetic field (blue curve: $k_0 = 0$).](image)
4.2. Results

4.2.1. Momentum measurement

With tracks generated in the non-uniform magnetic field, we compared the reconstructed reciprocal momentum and confidence level distributions for the track-fit results using the original helical track model assuming a uniform field and those using the segment-wise helical track model in Fig.6. The total momentum is calculated as \( p = p_t \sqrt{1 + \tan^2 \lambda} \), in which \( p_t \) and \( \tan \lambda \) are the fitted parameters in the state vector. In Fig.6(a), we used the original helical track model with a uniform magnetic field: \( B = (0, 0, 3 \, \text{T}) \) to reconstruct the tracks. The distribution of the reciprocal momentum on the left panel of Fig.6(a) is non-gaussian, and the confidence level distribution on the right panel has a delta-function-like peak at zero, indicating that the fitting is inconsistent with the simulated tracks. This is exactly why the simple helical track model used in the original KalTest must be updated for the non-uniform magnetic field situation for future linear collider experiments.

The result reconstructed with the segment-wise helical track model assuming the same magnetic field as that of the track generation is shown in Fig.6(b). The mean value of the momentum distribution on the left panel is very close to the expected, 0.1 (GeV/c)^{-1}, and its standard deviation is consistent with Gluckstern’s formula[7]. The flat confidence level distribution on the right panel implies the fitted tracks and hits are also consistent. Therefore, from Fig.6(a) and 6(b), it can be concluded that the improved algorithm with the segment-wise helical track model performs much better in track reconstruction since the non-uniformity of the magnetic field is properly taken into account.

In fact, the bad resolution in Fig.6(a) was found to come from a momentum bias. This is because the magnetic field becomes weaker as \( z \) increases, although it is assumed to be constant in the reconstruction with the original helical track model, thereby overestimating \( p_t \). Since the magnetic field decreases with the \( z \) coordinate, as the dip angle increases, the reconstructed \( 1/p_t \) decreases.

The results for different non-uniformities and dip angles with a track step size (i.e. helix segment size) of 6 mm are given in Table 1(a). As expected, for a specified dip angle, the momentum bias becomes more prominent as the non-uniformity increases, since the assumption that the magnetic field is approximately uniform between two nearby layers becomes inappropriate.
Figure 6: Comparison of track-fit results with the original helical track model and those with the segment-wise helical track model for tracks generated in a non-uniform magnetic field. The tracks were generated with the coefficient $k_0$ of the non-uniformity being 1, and the momentum 10 GeV/c.
Table 1: Mean and RMS of $1/p$ (in units of $10^{-1} \cdot \text{(GeV/c)}^{-1}$ and $10^{-5} \cdot \text{(GeV/c)}^{-1}$, respectively)

(a) Step size 6 mm

| $k_0$ | $\lambda = 0.1$ | $\lambda = 0.3$ | $\lambda = 0.5$ |
|------|----------------|----------------|----------------|
| 0.5  | 1.0000/8.02   | 0.9999/7.79    | 0.9998/7.34    |
| 1    | 1.0000/8.03   | 0.9998/7.89    | 0.9995/7.65    |
| 2    | 1.0000/8.05   | 0.9997/8.09    | 0.9990/8.36    |
| 3    | 0.9999/8.07   | 0.9999/8.31    | 0.9984/9.20    |

(b) Step size 1 mm

| $k_0$ | $\lambda = 0.1$ | $\lambda = 0.3$ | $\lambda = 0.5$ |
|------|----------------|----------------|----------------|
| 0.5  | 1.0000/8.02   | 1.0000/7.79    | 1.0000/7.34    |
| 1    | 1.0000/8.03   | 1.0000/7.89    | 0.9999/7.65    |
| 2    | 1.0000/8.05   | 0.9999/8.10    | 0.9998/8.36    |
| 3    | 1.0000/8.07   | 0.9999/8.32    | 0.9997/9.21    |

By the same token, the bias grows with the dip angle. Table 1(b) tells us that
the mean value can be improved by adding dummy stepping layers to make
the helix segment short enough and consequently the algorithm is forced to
reevaluate the magnetic field at more stepping points. Tables 1(a) and 1(b)
for the different step sizes have consistent momentum resolutions and as the
non-uniformity becomes large the momentum resolution also increases be-
cause the average magnetic field experienced by the tracks becomes smaller.
This explanation is also valid for the dip angle dependence for a fixed $k_0$
value (2 or 3). For the low non-uniformity values (0.5 or 1), the momentum
resolution is dominated by the lever arm length and hence as $\lambda$ increases, the
transverse momentum gets smaller, and the resolution gets better.

4.2.2. CPU time

The time consumption of relevant functions in the track fitting (without
the dummy layers to force the stepping size to be 1 mm) for 1000 tracks is
listed in Table 2. These CPU times were measured on a laptop PC with a
2.4 GHz Intel Core 2 Duo processor. The sum of CPU times consumed by
track propagation and filtering turned out to essentially be the total track
fitting time. From Table 2, one can see that the time consumption increased
Table 2: Time expense of relevant functions with the segment-wise helical track model for 1000 tracks in units of sec..

| Function                      | Time expense |
|-------------------------------|--------------|
| Total                         | 18.82        |
| TVKalState::Propagate         | 11.53        |
| TVKalSite::Filter             | 7.27         |
| TTrackFrame::TTrackFrame      | 0.87         |
| TTrackFrame::TransformVector  | 6.59         |
| TTrackFrame::TransformSv      | 2.58         |
| TVSurface::CalcXingPointWith  | 5.90         |

by about 10 seconds because of using the track frame transformation class, which was absent in the original KalTest. This means that the CPU expense was approximately doubled by the introduction of the segment-wise helical track model. The functions transforming vectors are called many times mostly in the function calculating the crossing points. One possibility to reduce this time consumption is to optimize the number of calls for the crossing point calculating function.

5. Summary

The helical track model in KalTest has been updated for the track fitting in a moderately non-uniform magnetic field by introducing the concept of segment-wise helical track model and implemented in the \texttt{THelicalTrack} class. The segment-wise helical track model assumes that the magnetic field is approximately uniform between two nearby measurement sites and approximates the track segment between the two sites to be a helix determined by the track momentum and the magnetic field at the starting site of the step. The non-uniformity of the magnetic field is then taken into account by reevaluating the magnetic field at the end site of the step for the next step. This requires the segment-wise helical track object \texttt{THelicalTrack} to carry a coordinate frame object \texttt{TTrackFrame} as its data member to specify the local coordinate system in which the state vector consisting of the helix parameters is defined. The helix frame is then transformed at each step so as to make the local $z$ axis pointing to the new magnetic field direction. The coordinate transformation modifies the original state propagator and its
derivative. In this paper we have elaborated the mathematical formulation of the modifications for Kalman-filter-based tracking. It should be emphasized that the modifications are localized in the THelicalTrack class and hence are invisible to the other part of the KalTest classes. This significantly facilitated the implementation of the code to handle the non-uniform magnetic field since there was essentially no need to touch the original KalTest architecture except for the track model THelicalTrack and its supporting class TTrackFrame. The algorithm should hence be relatively easy to port to any other Kalman-filter-based track fitting package.

We tested the segment-wise helical track model and its performance in Kalman-filter-based track fitting in a non-uniform magnetic field. The test showed that the track fitting with the segment-wise helical track model works very well for a modest field non-uniformity and yields correct track momentum values in the non-uniform magnetic field. It was also demonstrated that the track fitting performance can be enhanced by adding dummy layers (stepping layers) to reduce the step size so that the track fitting works in a highly non-uniform field situation.

The CPU time expense was measured and found to be approximately doubled as compared to the original KalTest for a uniform magnetic field. The increase was mostly due to the repeated coordinate transformations for various objects in stepping, calculations of crossing points of the track and measurement layers in particular. It can probably be improved by optimizing the code for the crossing point calculations.

The source code of the new KalTest with the segment-wise helical track model can be downloaded from the web page of updated KalTest.

Acknowledgment

The original KalTest package was developed in the framework of the LCTPC collaboration. We would like to thank all the developers including K. Hoshina, Y. Nakashima, and A. Yamaguchi for their significant contributions in the early stage of the development. The authors are grateful to the ILD software team and the members of the LCTPC group for useful discussions and supports for this work. Among them Steve Aplin and Frank Gaede deserve special mention. Their efforts to integrate the original KalTest package into the standard ILD event reconstruction chain and to test it thoroughly in various simulation studies set a firm basis on which this work could be built. The first author would like to thank the support from ILC group of
Institute of Particle and Nuclear Studies, KEK and LCD group of CERN. He also thanks to useful advice from Martin Killenberg. This work is supported in part by the Creative Scientific Research Grant No. 18GS0202 of the Japan Society for Promotions of Science (JSPS), the JSPS Core University Program, and the JSPS Grant-in-Aid for Science Research No. 22244031, and the JSPS Specially Promoted Research No. 23000002. This work is also supported in part by National Natural Science Foundation of China under Contract Nos. 11075084, 11161140590.

Appendix A. The modified propagator matrix

The concrete form of $F_{k}^{\text{rot}}$ in Eq.(25) is shown here. Notice first that at each step before the transformation to the new frame, the pivot of the predicted state vector is temporarily taken to be the predicted intersection of the track with the measurement layer there, implying that $d_\rho$ and $d_z$ are zero.\footnote{After the frame transformation the pivot is moved to the actual hit position from the predicted intersection.} According to Eq.(21), the momentum calculation needs only the three non-zero parameters of the state vector. Now let the state vector $a' = (\phi_0, \kappa, \tan \lambda)^T$, then the calculation of the first of the three derivatives in $F_{k}^{\text{rot}}$ is straightforward:

$$
\frac{\partial p}{\partial a'} = \begin{pmatrix} 
-\frac{1}{|\kappa|} \cos \phi_0 & s_\kappa \sin \phi_0 & 0 \\
-\frac{1}{|\kappa|} \sin \phi_0 & -s_\kappa \cos \phi_0 & 0 \\
0 & -\frac{s_\kappa}{\kappa^2} \tan \lambda & \frac{1}{|\kappa|}
\end{pmatrix}.
$$

(A.1)

According to Eq.(22), the second of the three derivatives in $F_{k}^{\text{rot}}$ is

$$
\frac{\partial p'}{\partial p} = \Delta R.
$$

(A.2)

If we just use the non-zero components in Eq.(23), the derivative of the new state vector with respect to the momentum is

$$
\frac{\partial a''}{\partial p'} = \begin{pmatrix} 
-\frac{p_y}{p_T} & \frac{p_x}{p_T} & 0 \\
-\frac{s_\kappa p_x}{p_T} & \frac{s_\kappa p_y}{p_T} & 0 \\
-\frac{p_z}{p_T} & \frac{p_y p_z}{p_T} & \frac{1}{p_T}
\end{pmatrix}.
$$

(A.3)
The product of the three matrices:

\[ M = \frac{\partial a'' \partial p' \partial p}{\partial p' \partial p \partial a'} \]

is a 3 x 3 matrix. To use it in Eq.(25), the elements with zero values should be restored, namely

\[ F_{k}^{\text{rot}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & M_{00} & M_{01} & 0 & M_{02} \\ 0 & M_{10} & M_{11} & 0 & M_{12} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & M_{20} & M_{21} & 0 & M_{22} \end{pmatrix}. \quad (A.4) \]

References

[1] T. Abe, et al., The International Large Detector: Letter of Intent, arXiv:1006.3396.

[2] J. Abernathy, L. Bo, P. Conley, K. Dehmelt, R. Diener, et al., Recent developments and status of MarlinTPC and related software, EUDET-MEMO-2010-07.

[3] S. Peter, Correction methods for tpc operation in inhomogeneous magnetic fields, LC-DET-2010-001.

[4] I. Antcheva, M. Ballintijn, B. Bellenot, M. Biskup, R. Brun, et al., ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization, Comput.Phys.Commun. 182 (2011) 1384–1385. doi:10.1016/j.cpc.2011.02.008.

[5] R. Fruhwirth, Application of Kalman filtering to track and vertex fitting, Nucl.Instrum.Meth. A262 (1987) 444–450. doi:10.1016/0168-9002(87)90887-4.

[6] K. Fujii, Extended Kalman Filter, http://www-jlc.kek.jp/subg/offl/kaltest.

[7] R. Gluckstern, Uncertainties in track momentum and direction, due to multiple scattering and measurement errors, Nucl.Instrum.Meth. 24 (1963) 381–389. doi:10.1016/0029-554X(63)90347-1.