

A Proposed Definition of `recob::Track` Trajectory Covariance Matrices

H. Greenlee

May 25, 2012

Contents

1	Introduction	1
2	Track Local Coordinate System	1
3	Use Cases	3
3.1	Momentum Vector Error Matrix	3
3.2	Impact Parameters and Vertex Reconstruction	4
4	Implementation	5

1 Introduction

Class `recob::Track` includes the following data members for defining the track trajectory.

```
std::vector<TVector3> fXYZ;           // Trajectory positions.  
std::vector<TVector3> fDir;          // Trajectory directions.  
std::vector<double>   fFitMomentum; // Trajectory momenta.  
std::vector<TMatrixD> fCov;          // Trajectory errors.
```

The definitions of the position vectors, direction vectors, and momenta are reasonably obvious (positions and directions are specified in the LArSoft global coordinate system, momenta in GeV/c, etc.). Up to now, the definition of the trajectory error matrix (`fCov`) has never been specified and there is no obviously preferred definition. The purpose of this document is to propose a definition of the trajectory error matrix.

2 Track Local Coordinate System

The minimum size of the trajectory error matrix is 5×5 , corresponding to a 5-dimensional track state vector on a surface. We consider the track state vector to

consist of the following five parameters: $(u, v, u', v', 1/p)$, where (u, v, w) are a track-local right-handed Cartesian coordinates and $u' = du/dw$ and $v' = dv/dw$. The (u, v, w) coordinate system is related to the global (x, y, z) coordinate system by a translation plus rotation. The first four track parameters specify the position and slope of the track on the surface $w = 0$.

A general rotation can be specified using three Euler angles. For our purpose (specifying the plane $w = 0$ with arbitrary orientation in space), it is sufficient to define a rotation matrix using two Euler angles θ and ϕ . We propose that the global-to-local rotation matrix $R(\theta, \phi)$ consist of a rotation by angle ϕ ($0 \leq \phi \leq 2\pi$) about the global x -axis, followed by a rotation by angle θ ($-\pi/2 \leq \theta \leq \pi/2$) about the local v -axis. Specifically,

$$R(\theta, \phi) = \begin{bmatrix} \cos \theta & \sin \theta \sin \phi & -\sin \theta \cos \phi \\ 0 & \cos \phi & \sin \phi \\ \sin \theta & -\cos \theta \sin \phi & \cos \theta \cos \phi \end{bmatrix}. \quad (1)$$

The choice of rotation matrix is partially arbitrary. Equation 1 is just one possible choice. Specifying a choice is the major content of this note.

The unit basis vectors of the track-local coordinate system in global coordinates are the rows of the rotation matrix R (Eq. 1).

$$\hat{u} = (\cos \theta, \sin \theta \sin \phi, -\sin \theta \cos \phi), \quad (2)$$

$$\hat{v} = (0, \cos \phi, \sin \phi), \quad (3)$$

$$\hat{w} = (\sin \theta, -\cos \theta \sin \phi, \cos \theta \cos \phi). \quad (4)$$

To be clear, the content of this proposal consists of the following two requirements regarding the track-local coordinate system.

- The local v -axis is in the global yz -plane.
- The local u -axis points in the $+x$ direction.

The full global-to-local coordinate transformation can be written as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = R \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}, \quad (5)$$

and inversely (local-to-global),

$$\begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} = R^T \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad (6)$$

where (x_0, y_0, z_0) is the origin of the local coordinate system.

The global-to-local transformations defined by Eqs. 1–6 are already implemented in class `TrackFinder/SurfXYZ`, which will make it convenient to implement in the Hit-based Kalman filter (it should also be easy to implement in `Genfit`, since `Genfit`

effectively allows the track-local coordinate system to be specified using all three Euler angles).

In momentum space, the same transformation equations obtain, except there is no translation.

$$\begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix} = R \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}, \quad (7)$$

and inversely,

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = R^T \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}, \quad (8)$$

A natural and mathematically well-behaved choice of track surface at any given trajectory point is the surface perpendicular to the track direction, in which the track state vector is $(0, 0, 0, 0, 1/p)$. This track surface is achieved by setting local origin (x_0, y_0, z_0) equal to the trajectory position \mathbf{fXYZ} , and setting the unit vector \hat{w} equal to the trajectory direction \mathbf{fDir} (solve for θ and ϕ).

3 Use Cases

This section gives several examples of how track parameters errors can be related to errors of quantities in the global coordinate system.

3.1 Momentum Vector Error Matrix

For arbitrary track surface, the momentum vector in the track-local coordinate system (assuming $p_w > 0$) is

$$p_u = \frac{pu'}{\sqrt{1 + u'^2 + v'^2}}, \quad (9)$$

$$p_v = \frac{pv'}{\sqrt{1 + u'^2 + v'^2}}, \quad (10)$$

$$p_w = \frac{p}{\sqrt{1 + u'^2 + v'^2}}. \quad (11)$$

In the special case of a track surface perpendicular to the track ($u' = v' = 0$), this reduces to $\mathbf{p}_L = (0, 0, p)$.

The error matrix of the momentum vector comes from the lower right 3×3 block $(u', v', 1/p)$ of the track error matrix. In order to transform errors from track parameter space to local momentum space, we need the Jacobian of the transformation of Eqs. 9–11.

$$\frac{\partial(p_u, p_v, p_w)}{\partial(u', v', p^{-1})} = \begin{bmatrix} \frac{\partial p_u}{\partial u'} & \frac{\partial p_u}{\partial v'} & \frac{\partial p_u}{\partial p^{-1}} \\ \frac{\partial p_v}{\partial u'} & \frac{\partial p_v}{\partial v'} & \frac{\partial p_v}{\partial p^{-1}} \\ \frac{\partial p_w}{\partial u'} & \frac{\partial p_w}{\partial v'} & \frac{\partial p_w}{\partial p^{-1}} \end{bmatrix}, \quad (12)$$

$$= \begin{bmatrix} \frac{p(1+u'^2)}{[1+u'^2+v'^2]^{\frac{3}{2}}} & -\frac{pu'v'}{[1+u'^2+v'^2]^{\frac{3}{2}}} & -\frac{p^2u'}{\sqrt{1+u'^2+v'^2}} \\ -\frac{pu'v'}{[1+u'^2+v'^2]^{\frac{3}{2}}} & \frac{p(1+v'^2)}{[1+u'^2+v'^2]^{\frac{3}{2}}} & -\frac{p^2v'}{\sqrt{1+u'^2+v'^2}} \\ -\frac{pu'}{[1+u'^2+v'^2]^{\frac{3}{2}}} & -\frac{pv'}{[1+u'^2+v'^2]^{\frac{3}{2}}} & -\frac{p^2}{\sqrt{1+u'^2+v'^2}} \end{bmatrix}. \quad (13)$$

When we specialize to the case where the track surface is perpendicular to the track ($u' = v' = 0$), the Jacobian simplifies to a diagonal matrix.

$$\frac{\partial(p_u, p_v, p_w)}{\partial(u', v', p^{-1})} = \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & -p^2 \end{bmatrix}. \quad (14)$$

We can use the above Jacobian to transform the lower-right 3×3 block of the track parameter error matrix to the momentum vector error matrix in the local coordinate system $\sigma_{\mathbf{p}_L}^2$.

$$\sigma_{\mathbf{p}_L}^2 = \begin{bmatrix} p^2\sigma_{u'}^2 & p^2\sigma_{u'v'}^2 & -p^3\sigma_{u'p^{-1}}^2 \\ p^2\sigma_{u'v'}^2 & p^2\sigma_{v'}^2 & -p^3\sigma_{v'p^{-1}}^2 \\ -p^3\sigma_{u'p^{-1}}^2 & -p^3\sigma_{v'p^{-1}}^2 & p^4\sigma_{p^{-1}}^2 \end{bmatrix}. \quad (15)$$

Finally, the momentum error matrix can be obtained in the global coordinate system using the rotation matrix R (Eq. 1).

$$\sigma_{\mathbf{p}}^2 = R^T \sigma_{\mathbf{p}_L}^2 R. \quad (16)$$

3.2 Impact Parameters and Vertex Reconstruction

Suppose that we have in the global coordinate system a vertex $\mathbf{v} = (x_V, y_V, z_V)$ and vertex error matrix $\sigma_{\mathbf{v}}^2$, and we wish to find the two-dimensional impact parameter and error matrix. This sort of calculation is most conveniently carried out in the track-local coordinate system. Therefore, we begin to rotating the vertex and vertex error matrix to the local coordinate system.

$$\mathbf{v}_L = R(\mathbf{v} - \mathbf{x}_0) = (u_V, v_V, w_V). \quad (17)$$

$$\sigma_{\mathbf{v}_L}^2 = R\sigma_{\mathbf{v}}^2R^T. \quad (18)$$

To find the impact parameter, we find the point of closest approach by propagating the track to the plane $w = w_V$. Since in the local coordinate system the track is propagating along the w -axis (nonmagnetic case), we know that the point of closest approach is $(0, 0, w_V)$, and the two-dimensional impact parameter in the w -plane is $(-u_V, -v_V)$.

The 2×2 error matrix of the impact parameter gets contributions from the error of the vertex and the error of the track. The error of the vertex is simply the upper left 2×2 block of the vertex error matrix in the local coordinate system.

To get the impact parameter error due to the track, the track error matrix needs to be propagated from the plane $w = 0$ to $w = w_V$. This propagation can be calculated exactly if propagation noise, interactions, and magnetic bend can be neglected

(probably a good approximation most of the time). If these things can't be neglected, then a track propagation utility can be used. The propagated (u, v) track parameters at the point of closest approach are (not assuming $u' = v' = 0$),

$$u_V = u + w_V u', \tag{19}$$

$$v_V = v + w_V v', \tag{20}$$

Propagating errors, the impact parameter error matrix due to track errors is

$$\sigma_{u_V}^2 = \sigma_u^2 + w_V^2 \sigma_{u'}^2, \tag{21}$$

$$\sigma_{v_V}^2 = \sigma_v^2 + w_V^2 \sigma_{v'}^2, \tag{22}$$

$$\sigma_{u_V v_V}^2 = \sigma_{uv}^2 + w_V^2 \sigma_{u'v'}^2 + w_V (\sigma_{uv'}^2 + \sigma_{vu'}^2). \tag{23}$$

4 Implementation

This proposal does not require any new data members be added to class `recob::Track`. Modules that make `recob::Track` objects will be required to fill data member `fCov` according to the definition of Sec. 2, which in practice means propagating or transforming track objects to the track surface defined in Sec. 2.

Class `recob::Track` will require some additional methods. At a minimum, `recob::Track` should provide a method to calculate and expose the rotation matrix R (and probably also R^T) as a `TMatrixD` object. Other methods that could be added include transformations of arbitrary vectors and error matrices between the local and global coordinate systems, the use cases described in Sec. 3, as well as methods to support other use cases that people may think of or find useful (vertex constrained tracks, kinematic fitting...). However, all use cases can also be implemented external to `recob::Track`, provided only that the rotation matrix R is available to the user.