1 Vertexing procedure: what for?

More precisely, what GLAST can/should gain from a vertexing step?

- **Gamma hypothesis**: If an event “looks like” a pair creation, the 2 resulting tracks should have a common origin. Consistency and noise rejection require that we make some quantitative comment on the “common origin” hypothesis.

- **Vertex localization**: not a big deal, i.e. we don’t really need it, do we? It still can be useful in helping an EventAnalysis step make the final decision:
  - Is the vertex far away from a converter?
  - Are the hits of the 2 tracks compatible with the vertex (missed layer?)

- **Evaluation of photon direction**: Critical, of course. Vertexing would provide the following (correct?):
  - Reevaluation of the momentum of each track at the vertex. Mandatory in the case of an helix, much less relevant for us: Reevaluation for a straight line is (should be) easy (trivial?). Besides, if we don’t bring in information from the calorimeter, we end up evaluating direction only
  - Calorimeter info., if available, would be easily accounted for by the vertexing.
  - Correct error propagation! In my mind it is the main quantitative gain. Am I wrong? Once we have error matrix for the track parameters, we need to combine them somehow... We need to explicitly account for the fact that they are constrained by a “same origin” requirement.

Technical Aparte:

- What is the current parametrization of a track? In the vicinity of a given reference point, I should need 4 parameters (5 for an helix)...
- Where is the track error matrix computed?

2 Vertexing procedure: the basics

The $\chi^2$ prescription is (more or less) the following:

1. Start with a set of tracks defined by their parameters $\vec{q}$ taken at some reference point.
2. the goal is to compute a vertex point $\vec{V}$ together with the tracks parameters $\vec{p}$ at $\vec{V}$. For that, consider the $\vec{q}$ to be functions of $\vec{V}$ and $\vec{p}$:

$$\vec{q} = \bar{F}(\vec{V}, \vec{p})$$

1
3. \( \chi^2 \) prescription si to minimize the following function:

\[
\chi^2 = \sum_{\text{tracks}} \Delta \bar{q}_i^T W_i \Delta \bar{q}_i
\]

where \( W \) is the weight matrix of a track and:

\[
\Delta \bar{q} = \bar{q} - \bar{F}(\bar{V}, \bar{p}),
\]

which is not 0 because the \( \bar{q} \) are known with errors.

4. In practice, to minimize the \( \chi^2 \) one needs to linearize \( \delta \bar{q} \). This is possible in practice, provided that it is performed in the vicinity of the vertex.

5. Linearization provides 2 matrices \( \frac{d \bar{F}}{d \bar{V}} \) and \( \frac{d \bar{F}}{d \bar{p}} \) with which solution is analytical (but of course for the possible necessity to iterate).

As a result, one of the important steps for vertexing concerns “propagation”:

- Choose an approximate vertex.
- Propagate the parameters back to a vicinity of the vertex.
- Propagate their error matrix.

Think of the following situation: one attempts to fit 2 tracks in GLAST. The first one has its first hit on layer 1, and the second on layer 2. Everything else being reasonable for these 2 tracks to come from the same vertex, one could/would/should use the first hit of the first track as an estimate of the vertex. Then, there is one converter between the first hit of the 2nd track and the region of the approximate vertex. In other words, careful propagation of the error matrix is most desired!

Technical Aparte:
One should design the vertexing code providing space for a propagator call...

3 The Perigee parametrization for a straight line

4 On Covariance matrix redefinition

Right now, CovMatrix return by Kalman filter doesn’t correspond to the expected parameters: for instance RayDoca uses (position,direction) to define track as a Ray object, before “vertexing”.

Is there agreement on the following:

1. After Kalman filter is done, we want to deal with straight lines \( \rightarrow \) smthg simpler than TkrFitTrack objects, but created out of it.
• Relevant parameters: Point and direction at first layer \( \rightarrow (x,y,z,\theta,\phi) \)
• Class TkrVtxTrack to interface to TkrFitTrack object without interfering/changing data members
• TkrVtxTrack might also have specific needed methods (more on that later)

2. Straight line (ie TkrVtxTrack object) is known within errors, given by the cov matrix of the Kalman step \( \rightarrow \) need to transform from \( (x,x_{slope},y,y_{slope}) \) to \( (x,y,z,\theta,\phi) \)

(a) First create \text{covMatrix} block diagonal with \text{covMatrix} C taking the 4 first rows/columns and \( \sigma^2_{zplane} \) at \( (5,5) \) (need to decide what error to use)

(b) Then in order: permutations 3\leftrightarrow 4, 2\leftrightarrow 3, 3\leftrightarrow 4 these are trivial SymMatrices. We end up with the parameters in order \( (x,y,z,x_{slope},y_{slope}) \)

(c) Transform \( (x_{slope},y_{slope}) \rightarrow (\theta,\phi) \): need to approximate by linearization:

\[
B = \frac{d}{d\theta} \begin{pmatrix} x_{slope} \\ y_{slope} \end{pmatrix} = \begin{pmatrix} (1 + \tan^2 \theta) \cos \phi - \tan \theta \sin \phi \\ (1 + \tan^2 \theta) \sin \phi \tan \theta \cos \phi \end{pmatrix}
\]

Then \( new\text{ConMat} = B \cdot old\text{CovMat} \cdot B^\top \).