FastMC Track Discussion

Notes from the ALCPG Sim/Reco meeting Nov 27, 2007

Notes by Rob Kutschke. Corrections and comments are welcome.

We discussed how to resolve problems with tracks generated by the FastMC. The root of the problem is the inconsistent use of the concept of a "reference point".

Jan's web page that Norman referenced is: http://confluence.slac.stanford.edu/display/~jstrube/A+New+Track+Interface
I have not yet had a chance to look at it so there is some chance that my notes below contradict it ....

Some external documentation:

1. After the meeting I found an old note from Tony that points to an LCIO document that defines the reference point: http://www-flc.desy.de/lcnotes/notes/LC-DET-2006-004.pdf
    - This has a good pedagodical description of the reference point.
    - But it is inconsistent with the org.lcsim sign convention for d0.
2. The document with the correct org.lcsim sign convention but a shorter discussion of the reference point is: http://lcsim.org/software/lcsim/apidocs/org/lcsim/util/swim/doc-files/L3_helix.pdf
3. Jan also has a document in progress: contrib/JanStrube/vertexing/vertexing.pdf
    - Tony: is there a way to make a link to this document?

The note from Tony also contains the definition of what is stored in the persistent representation of an LCIO track: From Rob Kutschke

This note is embedded in a broader discussion about tracks rooted at: Evolving the LCIO Track Class

As I understand it, our working proposal is:

1. All tracks created by the fast MC should be compliant with the track parameter definition in the apidocs for the getTrackParameters() method of the Track interface in org.lcsim.event.
2. Currently, for the fast MC, the reference point is the MC true production vertex of the track. This is poorly documented and has caused trouble when people were unaware of it.
    - Tim Barklow said that he was pretty sure that the code he used a year ago did not behave this way. He will check.
3. We propose to lift the confusion by defining the reference point to be (0,0,0) for all tracks, regardless of the location of the production vertex.
4. The track reference point is not a point on the track. It is the point against which the 5 helical parameters are defined.
5. I interpret the above to mean that Track.getMomentum() should return the momentum of the track at the point of closest approach to the z-axis
    - = point of closest approach to the origin in the xy plane.
    - Again this is true even if the track has a production vertex far from (0,0,0).
6. A corollary of 5 is that if I get many tracks that have a true production vertex at the IP, then they will all have different values of Track.getPosition() (different in all 3 coordinates).
7. The track interface has no method that returns the (x,y,z) coordinates of the position of the point of closest approach to the z axis.
    - We need to define how people get this info.
    - I think that Jan's Track class has a method to do this.
8. Tony suggested that a long time ago that the decision was made that getMomentum would return the values at the point of closest approach of the 3D trajectory to the 3D origin (0,0,0).
    - This is not the same point as that defined above.
    - This is certainly possible, but it seems odd to have gone through all the work of defining what the 5 helical track parameters mean and then having getMomentum() do something totally different.
    - Jan says that his code has the features necessary to return this information and also to return the position at this point.
    - I don't remember if Jan said anything about the error matrix at this point.
9. This leaves the issue of what to do with tracks that come from production vertices outside of the beam pipe. Norman proposed:
    - Define helical track parameters relative to the above reference point.
    - Smear them using the available parameterized covariance matrices.
    - This is not ideal since the parameterized covariance matrices presume that measurements are present in all layers and that the track originates inside the beam pipe.
    - We do not currently have the tools to do better.
10. To make his happen requires modifications to the fastmc code. I don't think we decided on who would make this change or if we know yet exactly what changes are necessary.
11. We need to hear back from Ron Casell and Tim Barklow how these proposed changes will affect their code, for better or worse.
12. At present there is no handy code to propagate covariance matrices away from the point of closest approach to the z axis.
    When we get this all working we need to summarize it in a working example that exercises the important features.
13. Caroline has kalman filter code that can address many of these issues and she would like us to evaluate it. I asked for a working example.
14. I need to understand the interaction of Tracks with ReconTrack, ReconstructedParticle and other Track-like objects