Subject: Re: PID package
Posted by StefanoSpataro on Thu, 06 Aug 2009 16:46:37 GMT
View Forum Message <> Reply to Message

Hello,

Quote:
First of all, as you wrote in the posting above, "this will replace the old PndLhePidMaker and the PndMicroWriter". As far as I understand you and also that what I see in the code, the new PndPidCorrelator is responsible for doing some PID (whatever this means) and for creating the complete list of RhoCandidates
(or PndPidCandidates) which is the input for the analysis part. Am I right?

Not exactly. I think you have missed to read the minutes of the EVO meeting on 1st July 09, where the structure was proposed and somehow accepted:

http://panda-wiki.gsi.de/cgi-bin/view/Computing/Minutes01Jul2009

The correlator is just correlating the track to pid detectors, and create candidates for pid. This are charged candidates (at the moment called PndPidCandidate), and neutral candidate. The correlator is not doing PID at all, just correlation.

Quote:
By looking a bit closer into the code I realized that the PndPidCorrelator does even more:
Apart from being the interface to the analysis part and for doing some PID related things also

The structure of the Candidate is not yet fixed, therefore neither the interface to the analysis. The starting point is just the old MicroCandidate. But the structure itself is still under discussion.

Quote:
c. an extrapolation of tracks to indiviual subdetectors (ToF, EMC, Mdt, DIRC, etc.) will be done and
d. detector specific PID related properties will be calculated there (e.g. truncated mean method for STT).
I think, this goes definitely far beyond that what should be done and should be provided in the PID related code. As you know the software has to be highly modular and flexible and has to make use of encapsulation in order to keep the code maintainable. This means that one has to decouple the different things which are currently done in the PndPidCorrelator and furthermore "non PID related" things should not be placed in the PID package at all.

Sorry but I have not understood at all this point.
Apart from correlation, dE/dx is calculated from the pid infos of the track. In theory my idea was to have single classes to retrieve useful informations from pid detectors. For complicated calculations like for cherenkov or tpc dE/dx this is my idea, but for simple calculations such as stt de/dx I have just implemented everything in the same class. Considering that (almost) nobody is working on pid detector informations, this makes life easier.

Quote:

In addition I have also some questions/remarks to technical points of the new code:

1. PndPidCorrelator:
 i. the implementation of the "singleton" has not been done properly. In case that you call the static method "PndPidCorrelator::Instance()" you will be get back a 0 pionter which could cause a crash in your application. In addition the constructor is defined there as "public" with the consequence that one can create such objects several times. There are a lot of documents available (on the web or books about design patterns) where one can find nice descriptions how to implement a singleton in a proper way.

The isntance is just a dummy function that is kept from old code, and that is never used. I could also removed it. PndPidCorrelator is a task, inheritated from FairTask and therefore from TTask, then all the basic functionality are there. It is not an object that everybody could use, but it has to be used inside our Run task list.

Quote: ii. in e.g. lines 758, 763,764 a division by zero has not been caught.

758 is already protected. In 763 and 764 the denominators are never zero.

Quote:
 iii. in line 436 a stack overflow has not been caught.

Sorry but I have not understood this(the code was coming from stt developers). What is exactly the error?

Quote:
2. PndPidCandidate:
 What I see there is that lots of specific properties (in general doubles or integers) are copied to this object via "set methods". Why should this class not have just references to the relevant reco objects? Besides a better performance (avoidance of additional cpu time for several hard copies and of increasing memory) this would keep the code more flexible and maintainable.
 E.g. If in the software as it is right now right some relevant methods in one reco class will be removed or changed, you have also to modify the PndPidCorrelator and the PndPidCandidates.
In case of just holding references to those (abstract) objects, nothing at all has to be changed in the PndPid* classes.

For all the detectors the index of the corresponding hit TCA are kept, then it is always possible to do what you want. The diea is that at some stage you do not store hits anymore but only the Candidate, which has all the useful informations. You copy one time the info but then you remove all the other stuff coming before.
Yes, if some function changes also the Correlator should change. This can be avoid using classes instead of all the GetXXXInfo(), but first there should be somebody working on those classes and taking care them.

Quote:
 Another improtant point is the access to the informations of the track objects. At the stage

where you create the PndPidCandidate it is still not clear whether it fulfills the requirements for a specific particle type like electron, pion, etc. Consequently you have to "hard copy" all relevant properties
for all possible particle types of this track object (i.e. 5 time covariant matrices, 5x momentum, 5x vertex, etc.).
In case of holding a reference to the track object all infos are automatically available in your candidate. One has access to all public methods of this object and therefore also to the covariant matrices related to the different particle types. Another point is that the reference to the track object would allow to refit the track in the analysis part.

It is the same discussion of the detector TCA. Just storing the PndTrack ID (which is not yet written because the not fixed structure of the candidate), one is able to retrieve the info. Another option could be to store the PndTrack itself inside the Candidate, so that refit is always possible without asking for (which TCA gave me the correct track?). In this case it would be easier to merge infos coming from different PndTrack, such as one for the barrel and another for the forward tracking.
But this is just a proposal.

Ideas and contributions about the Candidate are welcome.

My idea is the have an abstract candidate (such as the VAbsCandidate), with the basic stuff (i.e. momentun, emc properties, and nothing else), and then two classes inderitated from it, Neutral and charged candidate. I.e., in charged candidate we need to store dE/dx, tof and so on, while this infos are useless for a neutral candidate, or at least a photon should not have mvd de/dx. But all this is still in fieri... the analysis structure should proceed together with the pid part, but at the moment it seems everybody is in holiday (me included )

Thanks for the comments, i hope I have clarified you most of the things. it is good that somebody else than "authors" takes a look and give opinions. Of course, if you want to write some classes, you are also welcome