Subject: PID package
Posted by StefanoSpataro on Mon, 03 Aug 2009 16:25:29 GMT
View Forum Message <> Reply to Message

Daer all,
the new version of the PID package is inside the trunk, as described some EVO meetings ago. This will replace the old PndLhePidMaker and the PndMicroWriter.

At the moment only the correlation part is implemented, filling the so called PndPidCandidate (not yet inherited from TAbsMicroCandidate).

If you want to try it, just check macro/pid, and add the run_pid_tpc.C macro after reco. You can use as starting track the prefit track from lhe (as in svn) or the refitted one by genfit. In this case, you have to modify one line and write "LheGenTrack" instead of "LheTrack":

PndPidCorrelator* corr = new PndPidCorrelator();
corr->SetInputBranch("LheGenTrack");

If you have other TCA made of PndTrack objects, you can also set the name of your TCA. The PidCorrelator is independent on the tracking algorythm.

Still the correlation code needs some improvements, but at least it seems to work.

Feedback is welcome

Subject: Re: PID package
Posted by Bertram Kopf on Thu, 06 Aug 2009 12:44:30 GMT
View Forum Message <> Reply to Message

Dear Stefano,
I took a look into the new version of the PID packages and tried to figure out what is going on there.
I do not understand the code in all details and I have, therefore, a couple of questions and comments on it.
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?
That means that this class should represent
a. the interface to the analysis (BTW: in my point of view such an interface is one of the most important part of the software) and
b. some PID related things/tasks

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

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.

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.
 ii. in e.g. lines 758, 763,764 a division by zero has not been caught.
 iii. in line 436 a stack overflow has not been caught.

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.
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.

Best regards,
Bertram.

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

---

Subject: Re: PID package
Posted by Bertram Kopf on Fri, 07 Aug 2009 09:52:54 GMT
View Forum Message <> Reply to Message

Hi Stefano,

Stefano Spataro wrote
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.

O.k. this means that the correlator should be only responsible for the collection of the properties of all subdetectors which might be useful for PID. When I look to your proposed data flow I come to the conclusion that it is only foreseen to collect all PID information, to apply a network etc. and based on this result to create a RhoCandidate. This means that just a part of the global PID will be supported. As you know most experiments make use of likelihood based algorithms where e.g. each detector provides PID likelihoods which will be combined

afterwards. Does it mean that subdetector PID will be skipped completely? Abstract classes, mechanisms for the possibility to choose different algorithms at runtime and common tools to handle probabilities etc. are - I think - also very helpful to provide.
I don't find any of such classes and tools in the present code. Is there an idea and a concept to implement such things?

Stefano Spataro wrote

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.

The track matching with the detectors STT, EMC, Tof and MVD are definitely done in this PndPidCorrelator. Such a track matching is in general detector dependent and not that easy and should be done separately. Does it mean that this is right now a workaround since essential parts in the reconstruction are missing? If so I am not sure if such a workaround - to put everything in one class/method - is the best solution. In my point of view life is easier to start with a proper design/structure and if something is still not there to provide dummy classes. With your workaround you have to change and reorder a lot of parts again and again.

Stefano Spataro wrote
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.

Then please remove the relevant lines immediately. Especially newcomers should not see such nonsense.

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

yes, you are right!

Stefano Spataro wrote
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?

sorry, I meant an overflow of the array. This must be caught!

Stefano Spataro wrote
For all the detectors the index of the corresponding hit TCA are kept, then it is always possible to do what you want.

Does it mean that this class provides just methods where one can ask for the index where to find the pointer in the branch? Does it mean that I have to take care of to build the transient object? This is very error prone because I have to know the correct tree and branch and in addition it is not type save. Isn't there a common mechanism available which is encapsulated from reconstruction?

Cheers,
Bertram.

Subject: Re: PID package
Posted by StefanoSpataro on Fri, 07 Aug 2009 14:04:29 GMT
View Forum Message <> Reply to Message

Hi Bertram,

Bertram Kopf wrote O.k. this means that the correlator should be only responsible for the collection of the properties of all subdetectors which might be useful for PID. When I look to your proposed data flow I come to the conclusion that it is only foreseen to collect all PID information, to apply a network etc. and based on this result to create a RhoCandidate. This means that just a part of the global PID will be supported. As you know most experiments make use of likelihood based algorithms where e.g. each detector provides PID likelihoods which will be combined afterwards. Does it mean that subdetector PID will be skipped completely? Abstract classes, mechanisms for the possibility to choose different algorithms at runtime and common tools to handle probabilities etc. are - I think - also very helpful to provide.

I don't find any of such classes and tools in the present code. Is there an idea and a concept to implement such things?

Youa re completely correct, and I try to explain the scheme the have thought. My PID slides could help.

The correlator is just doing correlation between tracks and detectors, creating the PndPidCandidate (neutral and charged). This object is storing all the indexes and almost all the useful informations whicl will be useful for PID, but it is not doing PID at all.
Once the "Candidate" is created, or better once all the candidates are created, then you can run the algorythms (still not yet implemented). The algorythms could be single detector PID (such as MVD dE/dx), or more complicated stuff such as multi variate analysis, neural network, and so on.
Each algorythm loops over the PidCandidates, and create a new TCA made of PndPidProbability objects, and with the name related to the algorythm. PndPidProbability is just a collection of probability of each candidate of being an electron/pion/kaon/proton/gamma/pi0/etc etc etc.
This means that you store one TCA for the candidate, many TCA for the probability. Each candidate is linked to the many algorythm TCAs, i.e. position 0 of the candidate coresponds to position 0 of the MVD dE/dx Probability and position 0 of the EMC shower propertied neural network, position 1 corresponds to .. etc etc..

Each candidates is linked to many kind of probability coming from different algorythms.

Once the user want to run some selection, he should use some TPidSelector, that can select paricles according even to combination of different algorythm.
I.e., please select me electrons with probability > 0.1 in EMC shower neural network algorythm, and pions with prob >0.1 combining DIRC algorythm and tpc dE/dx. Just an example.
Of course, once the code wil be fixed, we will provide "standard" pid selectors, so that everybody will use the same stuff without increasing the amount of "entropy" in the analysis.

In this way I think and hope we can have all the features and the flexibility you are correcting asking for (but better ideas are always welcome)

Quote:
The track matching with the detectors STT, EMC, Tof and MVD are definitely done in this PndPidCorrelator. Such a track matching is in general detector dependent and not that easy and should be done separately. Does it mean that this is right now a workaround since essential parts in the reconstruction are missing? If so I am not sure if such a workaround - to put everything in one class/method - is the best solution. In my point of view life is easier to start with a proper design/structure and if something is still not there to provide dummy classes. With your workaround you have to change and reorder a lot of parts again and again.

I have thought of separating correlation and "information" extractions. You are correct, but considering that at the moment I am the only one working on this code and that alone there are many things to do, I prefered to leave the PndLhePidMaker structure, in order having (almost) everything running and ready for the next levels (pid algos and selection). I plan to de-encapsulate single parts once they will be ready. I think I will start with the muon class, which I would like to write.

Quote:
Then please remove the relevant lines immediately. Especially newcomers should not see such nonsense.

I will do it, however users should not play with task but only with data objects. This si the

reason why I am slow in deleting this things.

Quote:
iii. in line 436 a stack overflow has not been caught.
(cut)
sorry, I meant an overflow of the array. This must be caught!

In theory endnum is 60% of sttCounts, which corresponds to the sixe of sttdedx. Then in theory it should be always lower than the array size. But I will take a look if maybe some exceptions could be nasty.

Quote:
Does it mean that this class provides just methods where one can ask for the index where to find the pointer in the branch? Does it mean that I have to take care of to build the transient object? This is very error prone because I have to know the correct tree and branch and in addition it is not type save. Isn't there a common mechanism available which is encapsulated from reconstruction?

I will start from the end: no, the common mechanism is not yet written. It could be useful, but at the moment I have not a clear idea on how it is possible to do this. Comments from the core experts are welcone. At the moment the "authors" should create the corresponding object activating the TClonesArray from the branch. The TCA in general are standard (MvdHit, SttHit, GemHit, TofHit, DrcHit, TpcCluster, EmcCluster, etc..), but for newcomers maybe this could raise some problems.
This point is still open.

P.S. I have already answered to this message but then the browser decided to close... this is the second time, I am not reading again, I hope it is clear enough

---

Subject: Re: PID package
Posted by Jens Sören Lange on Tue, 11 Aug 2009 11:03:42 GMT
View Forum Message <> Reply to Message

Dear Bertram et al,

I would like to add a few more notes, because I think that it is an important and interesting discussion,
and I have a different opinion about some of the issues which you raised.

1.) The "set methods".
They are by intention.
It was one of the PandaRoot principles from the beginning.
It is intended to be a protection mechanism, that one class is not
able to simply overwrite data members in a another class
(example "the pT of my track is negative, which package did that?")
Or, in other words, the avoid "global variables".

2.) It seems you would like to strictly separate
the reco part (track fitting) on one side
and the PID part (e.g. track extrapolation) on the other side
(and then assign PID into analysis instead of reco).
And your point seems to be (please correct me, if I'm wrong),
that you think that Stefano's approach (i.e. the track extrapolation
and filling PID information directly after the track fitting,
i.e. in the reco part) is in your opinion not a good approach.
So, what you have in mind, is
a.) tracks as reco objects
and
b.) tracks as PID (in analysis) objects,
and in most cases they are probably the same.
Now you are proposing to work with references to the track objects.
However, what then?
-> do you want to modify the tracks then in the PID part?
or, in other words, overwrite the tracks?
If yes, I would strongly vote for "hard copying" all the tracks.
Sure, as you say, these are a lot of data and that was actually
my counter argument when you brought up your proposal
(of the separation of reco and PID)
on the tracking hands-on meeting at GSI in March 09.
It is not only the py,py,pz, but vertex, covariance matrix, etc. etc.
But I really think it would be safer to copy the tracks,
and not overwrite.
That is why I think that Stefano's approach
is quite nice (filling the TCandicates _before_ you copy)

3.) Coming back to the point of probably overwriting tracks.
The question would be, why.
It seems that you would like to do a track fitter refit
in the PID part (i.e. then at a quite late stage in analysis).
But we decided on the PID Mini-Workshop at GSI in Sep 2007 that
track fitting will be done for 5 different masses (pi,k,p,e,mu)
anyhow.
That's also the Belle approach, and it was a majority vote to use
this as PandaRoot PID approach at that workshop.
That's even at a step before the PID.
The PID then tries to make a decision, which particle it was,
but using the momentum from the (already before finished) track fitting.
Probabilities (e.g. the probability that THIS track is a pion)
should then be calculated using the momentum from the track fitted
with THAT pion mass hypothesis.
What you probably have in mind (if I understand correctly),
is improving the PID decision by improving the track fitting,
but very late in the reco sequence, after track fitting
has already been performed, using some new information (from PID).
a.) But which new information could improve the fit?
(my point is: maybe I cannot see exactly the need for another
track refit in PID). Track Refit is important after final alignment
and/or IP vertex information from database etc. etc.
but is it really useful after calculating the PID probabilities?

Even dE/dx (which could enter the fit as a weight for hits)
is already fixed and known before the PID part.
b.) I would also vote for avoiding a (sort of) "hidden" (re-)track fitting
in the PID part. (or, in other words, I would think that a track fitting
should only be done in the tracking part, i.e. in reco).

4.) a re-fit is still possible for the TCandidates.
That's for example the vertex constraint fit or the mass constraint fit
as Klaus and Dipak implemented it anyhow (shown e.g. in
the Torino tutorial).

cheers, Soeren

Subject: Re: PID package
Posted by Bertram Kopf on Wed, 12 Aug 2009 13:36:07 GMT
View Forum Message <> Reply to Message

Hi Soeren,

Jens Soeren Lange wrote
1.) The "set methods".
They are by intention.
It was one of the PandaRoot principles from the beginning.
It is intended to be a protection mechanism, that one class is not able to simply overwrite data members in a another class
(example "the pT of my track is negative, which package did that?")
Or, in other words, the avoid "global variables".

I don't see any protection mechanism by using hundreds of set methods. To provide such set methods speaks completely against the encapsulation of the data members. It is possible to manipulate all private data members from outside by using these set functions. The encapsulation of the data member is one of the basic principles in oo programming. Of course, in some specific cases one can make use of set functions, but for the initialization of an object one should/has to use the constructor.

Jens Soeren Lange wrote
2.) It seems you would like to strictly separate
the reco part (track fitting) on one side
and the PID part (e.g. track extrapolation) on the other side
(and then assign PID into analysis instead of reco).
And your point seems to be (please correct me, if I'm wrong),
that you think that Stefano's approach (i.e. the track extrapolation
and filling PID information directly after the track fitting,
i.e. in the reco part) is in your opinion not a good approach.
So, what you have in mind, is
a.) tracks as reco objects
and

b.) tracks as PID (in analysis) objects,
and in most cases they are probably the same.
Now you are proposing to work with references to the track objects.
However, what then?
-> do you want to modify the tracks then in the PID part?
or, in other words, overwrite the tracks?
If yes, I would strongly vote for "hard copying" all the tracks.
Sure, as you say, these are a lot of data and that was actually
my counter argument when you brought up your proposal
(of the separation of reco and PID)
on the tracking hands-on meeting at GSI in March 09.
. . .

No, the track fitting, the track matching as well as the PID is definitely part of the reconstruction. What I wanted to point out is that one has to disentangle these different things within the reconstruction. It is of course not a good idea to do all these things within one method.
Concerning the track object: I don't want to have different track objects for one track. One, and only "one" abstract track object. The only point is to get access to all public functionalities by using references to this object, instead of hard copies of specific informations. To avoid manipulations on this object one can define it as a constant reference.

Jens Soeren Lange wrote
3.) ...
It seems that you would like to do a track fitter refit
in the PID part (i.e. then at a quite late stage in analysis).
But we decided on the PID Mini-Workshop at GSI in Sep 2007 that
track fitting will be done for 5 different masses (pi,k,p,e,mu)
anyhow.
...
That's even at a step before the PID.
The PID then tries to make a decision, which particle it was,
but using the momentum from the (already before finished) track fitting.

I don't want to do a re-fitting in the PID part. I totally agree with you that one should do track fitting for all 5 hypotheses (at least for low momenta particles where one expects different results). This, we have also done in the reconstruction for all our Physics Book studies. And, of course, at a later stage of the reconstruction the PID should be done. BTW: The track fitting probability for the different mass hypotheses could also be a helpful PID information.

Jens Soeren Lange wrote
4.) a re-fit is still possible for the TCandidates.
That's for example the vertex constraint fit or the mass constraint fit
as Klaus and Dipak implemented it anyhow (shown e.g. in
the Torino tutorial).

What you meant here is the vertex fitter which makes only use of the obtained track parameters and covariance-matrices obtained by the global track fitter. This has completely nothing to do with the track fitter. For the vertex fitter you combine different tracks (at least two)

and try to find a common vertex. What I meant instead is the re-fitting of the tracks in the analysis part. This would make sense when e.g. the alignment, calibration etc. have been improved. In the Physics Book software for example it is possible to switch between a cache and a refit mode where parts of the reconstruction can be re-done.

Cheers,
Bertram.