Subject: Strategy discussion on Track objects
Posted by Ralf Kliemt on Wed, 29 Apr 2009 12:05:14 GMT
View Forum Message <> Reply to Message

Hello everyone,

Currently we have several track objects:
Quote:genfit: Track, TrackCand
pnddata: PndTrack, PndTrackCand
lhetrack: PndTpcLheCMTrack (monte carlo track?), PndTpcLheTrack, PndLhePidTrack
riemannfit: PndRiemannTrack, PndTpcRiemannTrack
trackbase: FairTrackPar
stt: PndSttTrack
gem: PndGemTrack
dch: PndDchTrack
fsim: PndFsmTrack
tpc: PndTpcClusterTrack
Now the question is how we can proceed.

My first suggestion on this is to get rid of the detector specific objects by using only PndTrack and PndTrackCand. Be reminded to make use of the detector numbering definitoins in pnddata/PndDetectorList.h.

The second point is more complex and is regarded to the unification of the track objects used by the finders and fitters in general. A point which was made is that a track object should be able to provide a propagation functionality. This however conflicts with the policy to have data classes only storing their data, as far as I think.

I would be glad to have some comments by the different experts on this. When it is needed to discuss this more directly I'll set up an evo meeting.

Kind regards, Ralf.

Subject: Re: Strategy discussion on Track objects
Posted by StefanoSpataro on Wed, 29 Apr 2009 12:50:16 GMT
View Forum Message <> Reply to Message

Hi,
just to comment the lhetrack "tracks":

PndTpcLheTrack is the basic track
PndTpcLheCMTrack is derived from PndTpcLheTrack, and is the track adding the info from the conformal space hits
PndLhePidTrack is derived from PndTpcLheTrack, and is the track adding the info from pid detectors.

The output of the kalman is the LheGenTrack TCA, made of genfit Track objects.

So, it is just one basic track (which could be moved to PndTrack), the remaining stuff is just inheritated + the genfit Track.

Subject: Re: Strategy discussion on Track objects
Posted by StefanoSpataro on Wed, 29 Apr 2009 13:24:10 GMT
View Forum Message <> Reply to Message

I think, another important point is how to correlate the information from the prefit track (in my case, PndTpcLheTrack), the fitted track (genfit track), the track containing the pid information (PndLhePidtrack), and the PndMicroCandidate.

In my case, from the prefit track PndTpcLheTrack I create the pid track, and the kalman track. But there is no relation between the Track and the PndLheTrack. What should one use to fill the MicroCandidate?

I mean, it should be important, I think, to design how these classes should cooperate. At the moment I have no clear ideas, comments are welcome.

Subject: Re: Strategy discussion on Track objects
Posted by asanchez on Thu, 30 Apr 2009 08:35:30 GMT
View Forum Message <> Reply to Message

Hi Stephano, maybe you can use you PndLhetrack to fill the start parameters for the kalman filter, after that the fitted track object, Track, can be used to fill the MicroCandidate. Take a look into the
PndTools directory Analysis tools PndMicroWriter class.

best regards
Alicia S.

Subject: Re: Strategy discussion on Track objects
Posted by StefanoSpataro on Thu, 30 Apr 2009 09:22:03 GMT
View Forum Message <> Reply to Message

Hi,
this can be done, this is not the main problem.

The first problem is where to put the pid information in between.
Should we start from the lhetrack or the genfit one to collect pid info? But we should not be compelled to use the refitted tracks, we should have the possibility to chose between the two kinds of objects, which are simply different classes..

And then, we should be able to fill the MicroCandidate from genfit tracks and even from not fitted tracks. But they are two different objects.

And how to correlate the refitted tracks to the prefit track, or to the pid track?

I mean, the solution is not so easy.

Subject: Re: Strategy discussion on Track objects
Posted by Bertram Kopf on Thu, 30 Apr 2009 11:43:33 GMT
View Forum Message <> Reply to Message

Hi,
you are discussing here typical software design issues. In my point of view it would be helpful to firstly work out a -more or less- proper design, followed by dummy implementations before one then starts with concrete implementations of alorithms etc.

In this discussion 3 different things are mixed-up:
1. tracking of charged particles,
2. PID and
3. the interface to the analysis (PndMicroWriter)

Here are just few remarks to these 3 points:

1. Tracking: Ralph is completely right that only one common track object should be used. The important thing is to think about how such an object should look like, what features shoult it have, if there is the necessity of abstraction, etc..

2. PID: PID has in priciple nothing to do with tracking of charged particles. It should also be possible to do PID for neutral particles: e.g. gammas, neutrons, merged pi0's, electromagnetic split-offs etc. I think, it is therefore better to decouple it from the tracking and to provide only PID specific software parts. Of course, the relevant objects should have an assoiciation to the corresponding tracks or EMC clusters.

3. The interface to the analysis should not be based on the tracks. The reconstruction should finally provide lists of particle candidates (neutrals, charged, etc.) which are then
the input for the analysis. For sure, these candidates should also provide references to the corresponding tracks or clusters including covariance matrices etc. so that also these informations are easily accessible in the analysis part. Therefore one should think about how such a candidate (RhoCandidate ?) should look like.

Best regards,
Bertram.

Subject: Re: Strategy discussion on Track objects
Posted by Anonymous Poster on Thu, 30 Apr 2009 11:54:56 GMT
View Forum Message <> Reply to Message

Hi,

I think Bertram is right about the separation of tracks and PID info.
You say that one should define what is needed from the track object. This is very true. If there

are requirements to the tracks which are currently not fulfilled by the genfit Track, we will try to add them. However, since we are using genfit for charged particle tracking, I think this is the way it should go. To make another track object, that will need to learn extrapolation capabilities in a similar way to genfit again, does not make sense.

So, again, if there is any additional features of the track which are needed, we will accommodate them (if possible of course) in genfit. But I also understand that there has been concern in the past that PANDAroot shall not depend on genfit. But as this is the only charged track reconstruction we have and we will have in the near future, there is also no way around it. I do not know how to resolve this contradiction.

Cheers, Christian

Subject: Re: Strategy discussion on Track objects
Posted by Mohammad Al-Turany on Thu, 30 Apr 2009 21:10:00 GMT
View Forum Message <> Reply to Message

Hi,

I think almost all of us agree about having a well defined interfaces and cleaning the implementation for the Tracks, PID, etc. We already discuss this in the last collaboration meeting and even start doing something about it! What I do not understand is the statement:

But I also understand that there has been concern in the past that PANDAroot shall not depend on genfit.

In fact it was you (Christian and Sebastian) who wanted to have Genfit separated from the whole Pandaroot from the beginning with the argument "PANDA experiment changed the framework twice and if this happened again GenFit should not be changed!"

In fact Track, PID, etc. belongs normally to the PndData package, which then can be used by all detectors and/or packages, and what Christian suggest is simply all detector and analysis packages should be dependent on GenFit instead of PndData! What would be the problem if GenFit uses PndData? Do you really still believe that PANDA will through every thing in PandaRoot away and keep GenFit?

Maybe we have to clarify very soon the situation with this GenFit, and if you really think it is so general and independent may be you should distribute it as a stand alone package and we shift it to the external packages, or if it is a Panda software which is integrated inside panda software then it has to follow the rules as all other packages.

regards

Mohammad

Subject: Re: Strategy discussion on Track objects
Posted by Anonymous Poster on Thu, 30 Apr 2009 23:33:17 GMT
View Forum Message <> Reply to Message

Hi,

it is very clear also to me that we need clearly defined interfaces and classes. I guess this is why Ralph started this discussion. So, I thought the interface of a useful track object is not yet available. If it were, we wouldnt be discussing it, or maybe there is a misunderstanding after all. So could you try to make clear what your suggestion is please. As I said, I will accommodate what is necessary in genfit. If it is the wish of our community that the genfit track objects can somehow be converted in general PandaROOT classes, we can do that.

The central point of the discussion is to my mind, whether the central track object needs track extrapolation capabilities or not. Could you please give your opinions on that?

I do not understand why you want to move genfit out of PandaROOT. Apparently there is a feeling that we are somehow not respecting some rules (quote from your mail). What exactly is that we are doing wrong? What are your suggestions to improve the situation?

And please excuse me for being mildly frustrated about the code development in the past in PANDA. It just isnt so nice to work many years on some software and then having to redevelop it in another framework. Of course I want to avoid this from happening again. I am glad for you that you are 100% sure about the future of PANDA computing. But please allow me to not share this feeling completely. The past has taught me to be a bit careful about this.

But I want to stress again that if there are wishes by this community that the genfit track object shall be convertible into some common class that we will do that. Just tell us what you need. However, we can not discuss to change the structure of the genfit track object, because the whole design of genfit is based on it.

From reading your email I certainly feel that there is some hostility in the air. It is a pity that this is the case. I think we are in a good situation. We have in PANDAroot a full working charged track reconstruction. This something that we for example never had for all of PANDA in the old framework. This is a great step forward. And just the question of a data interface shouldnt split our community. We all have a common interest. And having track reconstruction is a part of the goal we all want to reach.

Cheers, Christian

Subject: Re: Strategy discussion on Track objects
Posted by Johan Messchendorp on Fri, 01 May 2009 14:01:11 GMT
View Forum Message <> Reply to Message

Dear all,

Let's try not to get too excited on these issues and let's try to avoid to revive old struggles and fights we had in the past. Important is that we on a very short term come with a scenario which is used and usable by us all. As Christian already said, we all have the same interest: a workable track reconstruction in our framework. So lets get back to the real issue, namely the point which Ralf stated earlier:

Quote:The second point is more complex and is regarded to the unification of the track objects used by the finders and fitters in general. A point which was made is that a track object should be able to provide a propagation functionality. This however conflicts with the policy to have data classes only storing their data, as far as I think.

I have to admit that partly this is a religious issue. So let me be nasty and refer to the computing model which states that one should try hard to minimize the number of internal dependencies. Furthermore, as a personal preference, I would like to avoid to be in a situation in which I need to load hundreds of libraries or find/include the two hundred corresponding header files in order to be able to look or perform a simple analysis on a root-file generated by a pandaroot. I, therefore, believe (!) one should try to separate as much as possible data classes from "functionality" issues, to keep it transparent at the end of the day. Ok, I have to admit, that I am also grown up with this tradition. So, if the majority of the tracking group has a different opinion, I wouldn't have any problem, though.

Kind wishes,

Johan.

Subject: Re: Strategy discussion on Track objects
Posted by StefanoSpataro on Fri, 01 May 2009 14:36:44 GMT
View Forum Message <> Reply to Message

Few comments from my side:

Bertram Kopf wrote Hi,
you are discussing here typical software design issues. In my point of view it would be helpful to firstly work out a -more or less- proper design, followed by dummy implementations before one then starts with concrete implementations of alorithms etc.

Indeed I am asking opinion about software design. The code and the algorythm is already implemented, working and tested with my personal design, wich is similar to what is used in HADES and CBM (if I have understood well): track (from tracking detectors) -> pidtrack (the object which stores all the informations for pid, the track + emc eloss, tof, etc.) -> microcandidate (the starting point for analysis). The selectors should be set somewhere, between pidtrack and micro candidate (which at the moment are almost similar) or after microcandidate.
I am just asking if there are ideas on how to have something better.

Bertram Kopf wrote
1. Tracking: Ralph is completely right that only one common track object should be used. The important thing is to think about how such an object should look like, what features shoult it have, if there is the necessity of abstraction, etc..

The object was designed some time ago, and now it exists also in the code: PndTrack (and PndTrackCand). One should just start to move the old objects to this new track class.

Bertram Kopf wrote
2. PID: PID has in priciple nothing to do with tracking of charged particles. It should also be possible to do PID for neutral particles: e.g. gammas, neutrons, merged pi0's, electromagnetic split-offs etc. I think, it is therefore better to decouple it from the tracking and to provide only PID specific software parts. Of course, the relevant objects should have an assoiciation to the corresponding tracks or EMC clusters.

I do not understand this point. Of course PID is different from tracking, but you can perform PID only if you have tracks. The PID object has momentum, and you ahve to use momentum + the other variables (such as emc eloss) to distinguish between particle. And only if you have tracks you can understand if one emc cluster comes from a photon or from a charged track. The, I would formulate this sentence as: PID depends on tracking. Or have I missed something?
And the track matching with pid detectors, which is a very important part of pid task, depends on track parameters. Even for the photon ID. The fact that the pid matching is done inside one directory called "lhetrack" is due to the fact that at the moment the algorythm is based on the LheTrack and not on the common PndTrack written one month ago.
The fact that genfit provides a track which will never be a base PndTrack (if I have understood well), complicates the history, because in this case the abstract class cannot be used. But I have an idea.
It could be possible to write a task, inside PndTools, which loops inside Track TCA, and creates a new TCA with PndTrack.This will be the standard track for correlation to PID detectors.
Of course the algorythm could be moved in a new pid package, once it will be uncorrelated to "local" lhetracks.

Bertram Kopf wrote
3. The interface to the analysis should not be based on the tracks. The reconstruction should finally provide lists of particle candidates (neutrals, charged, etc.) which are then
the input for the analysis. For sure, these candidates should also provide references to the corresponding tracks or clusters including covariance matrices etc. so that also these informations are easily accessible in the analysis part. Therefore one should think about how such a candidate (RhoCandidate ?) should look like.

I think this is already inside the rho package. The user needs just momentum and pid. But the user should be able also to perform his pid, or better to perform the standard pid for his channel, which could be different from another channel.

Ideas on class design about this tracking+pid+analysis are welcome. My personal idea is already inside the code, but I would like to listen to other opinions.

Subject: Re: Strategy discussion on Track objects
Posted by StefanoSpataro on Fri, 01 May 2009 14:40:50 GMT
View Forum Message <> Reply to Message

Hi,
I have forgotten the last comment.

The tool for track propagation should be geane. The object should not provide this poddibility, even because the data object should not care about magnetic field and geometry.
My opinion.
In each case PndTpcLheTrack has inside a simple propagation (using helix) which does not use field at all. This could be written as independent class.

Subject: Re: Strategy discussion on Track objects
Posted by Anonymous Poster on Fri, 01 May 2009 16:08:26 GMT
View Forum Message <> Reply to Message

Hi,

I have some question: Does PndTrack have extrapolation capabilities? If not, who is going to add them and how? Or maybe this is not what PANDAroot wants? I still dont get this point clearly.

About the PndTrackCand: How does it work? Does it hold indices to the hits that are used in this track? In genfit we have a TrackCand, which basically holds 2 standard vectors of integers: One for detId of the hit and one for the index in the corresponding TClonesArray of hits (e.g. PndTpcClusters or some MCpoints). What information is in PndTrackCand? I am sure we can convert it.

What do I need to do to make PndTrack objects. Is there just some constructor which takes position, momentum, etc.? I could implement a method in the Track object of genfit which returns such a PndTrack.

Cheers, Christian

Subject: Re: Strategy discussion on Track objects
Posted by Anonymous Poster on Fri, 08 May 2009 14:43:16 GMT
View Forum Message <> Reply to Message

Hi,

I dont know why this discussion wasnt continued. Are there simply no answers to my last questions?

Cheers, Christian

Subject: Re: Strategy discussion on Track objects
Posted by Ralf Kliemt on Fri, 08 May 2009 15:09:27 GMT
View Forum Message <> Reply to Message

Hello Sebastian and all others,

As I see the PndTrack and PndTrackCand it has to be developed according to our needs.

I think of the propagation issue with mixed feelings. Introducing the propagation capability along with the PndTrack class would be the easiest way for the user end. On the other hand I cannot judge how often that feature will be needed outside the Kalman code. There could be a Toolbox class, serving with the necessary things (like said propagation, even with different possible propagators?). The same is with the PndTrackCand, there might be tools needed to access the data.

I think it is a good policy to keep data and functionality apart from each other.

I'd like to see the others opinions that we find an agreement.
(It is a fundamental, but only technical thing.)

Ralf.

Subject: Re: Strategy discussion on Track objects
Posted by Sebastian Neubert on Fri, 08 May 2009 17:41:00 GMT
View Forum Message <> Reply to Message

Dear colleagues,

as far as I understand from your discussion there are several interests that should be fulfilled (and where I have the feeling everybody agrees):

1) There should be a central PndTrack object which is part of PANDAROOT and provides a unified interface between the track-reconstruction and the particle data classes which will be used in analysis code.

2) In order to do fitting and refitting we need a TrackPropagator. The standard solution here is GEANE but it might be interesting to be able to exchange that propagator. There needs to be an interface to pass information forth and back between PndTrack and the propagators.

3) We need fitting routines which have access to residual information between hits and tracks. We need to be able to transfer hit-data and track-data (and track propagation!) from and to the fitting routines - this means the data has to be put into a form that is understood by the fitting framework.

GENFIT provides solutions for point 2) and 3) namely the AbsTrackRep and AbsRecoHit interface classes and the Track class.

Now from my perspective the real conflict in the different possible solutions here is the following:

Should PANDAROOT use the fitting packages (GENFIT) or should GENFIT use the PANDAROOT structures?

I understand that Mohammad defends the latter position as this guarantees that condition

number one will be fulfilled, namely a central, unified access point for track data. Furthermore in that way PANDAROOT would not become too dependent on GENFIT.

For GENFIT we were forced to develop the data structure as it is now in order to provide its flexible functionality. These structures are deeply self-consistent which is why Christian is so reluctant to change it (and I strongly support him in this point).

In order to unify all this I would propose the following:

A "translator" has to be written which translates PndTrack into GENFIT language (Track) and the other way round. This would allow us to use the full GENFIT functionality while keeping mutual dependencies minimal (as required in the Computing Model). In software design pattern language this is an "Adapter".
Christian already has agreed to take over what is necessary from the GENFIT side to do this. It would be PANDAROOT responsibility to provide all information GENFIT needs to do its job.

The following scenario would be ideal for me:
One would use the Adapter to initialize GENFIT in the reconstruction, do the Kalman Fit and use the Adapter again to convert to PndTrack and store this object.

Then you take home your root file. In your analysis script you use a different Adapter to initialize the YAFF* and do a vertex-refit and convert back to PndTrack (+ related objects). You store the result.

Your colleague takes the refitted tree and uses the GENFIT Adapter to perform an extrapolation from the vertex to a certain detector with GEANE.

Best Regards, Sebastian.

*) Yet Another Fitting Framework

Subject: Re: Strategy discussion on Track objects
Posted by Anonymous Poster on Mon, 11 May 2009 09:39:21 GMT
View Forum Message <> Reply to Message

Hi,

could somebody please provide with a description on how to use the PndTrack* classes?

Especially: What exactly does the hit ID in the PndTrackCandHit stand for?
And what should I put for rho?

I am willing to do the job, but I simply cant do it until I gut the information I need.

Cheers, Christian

Subject: Re: Strategy discussion on Track objects
Posted by Tobias Stockmanns on Mon, 11 May 2009 09:43:08 GMT
View Forum Message <> Reply to Message

Hi Christian,

the hitID stands for the index of the Hit in the TClonesArray. Which TClonesArray to use is decided by the detID.

Rho was introduced as an ordering parameter, which allows a sorting of the hits along the track.

Cheers,

Tobias

Subject: Re: Strategy discussion on Track objects
Posted by Anonymous Poster on Mon, 11 May 2009 09:48:49 GMT
View Forum Message <> Reply to Message

Hi,

OK, and what should I use for rho?

Also: What is the meaning of the first and last track param in PndTrack?

Cheers, Christian

Subject: Re: Strategy discussion on Track objects
Posted by Tobias Stockmanns on Mon, 11 May 2009 10:42:09 GMT
View Forum Message <> Reply to Message

Hi Christian,

I think at the moment you can leave rho open. Later on it should be a parameter which allows the sorting of points within a track. For example the arc length.

First and last track param are the parameters of the track at the first and the last point within the track.

Cheers,

Tobias

Subject: Re: Strategy discussion on Track objects
Posted by Anonymous Poster on Mon, 11 May 2009 21:02:36 GMT
View Forum Message <> Reply to Message

Hi,

in order to make a PndTrack I need to give FairTrackPar for the first and last points. I looked into these files. First I would like to point out that in the header file the constructor is declared as

```
  /** Constructor with all variables **/
  FairTrackPar(Double_t x,  Double_t y,  Double_t z,
               Double_t fx, Double_t fy, Double_t fz, Double_t q);
```

And then in the cxx file it is defined as

```
FairTrackPar::FairTrackPar(Double_t x, Double_t y, Double_t z,
                           Double_t px, Double_t py, Double_t pz, Double
_t q)

  : fX (x),
    fY (y),
    fZ (z),
       fDX (x),
    fDY (y),
    fDZ (z),
    fPx (px),
    fPy (py),
    fPz (pz),
       fDPx (px),
    fDPy (py),
    fDPz (pz),
    fQp (0),
       fDQp (0),
       fq  (q)
{
```

This different naming of variables is strictly speaking not forbidden in C++, but it is very confusing. Please correct it to clarify.
I guess the paramters px,py,pz in the cxx file mean the momentum of the particle. But what about all errors and covariances? In this class there arent any data fields for covariances.

Am I doing something wrong here? Am I supposed to use some derived class? Please clarify this.

Cheers, Christian