Subject: [FIXED] Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by StefanoSpataro on Wed, 18 Sep 2013 21:10:18 GMT
View Forum Message <> Reply to Message

Hi tracking guys,
I see a potential bug, or maybe a feature, of the ideal track finders, i.e. PndFtsTrackerIdeal and PndSttMvdGemTrackingIdeal.

If I check PndFtsTrackerIdeal, first it search for FTS hits in the event:

```
if(fHits[0]->GetEntriesFast() == 0) {
  return;
```

If there are FTS hits, it start to fill the PndtrackCand with hits from MVD, GEM and FTS.
If there are at least 3 hits:

```
if( tcand->GetNHits() < 3 ) continue;
```

then the candidate is stored.

The main problem that I see is that there is no requirement that in the trackcand there is at least a FTS hit. For single particle events it is the same. But if you have a 2-particle events, one particle in the forward part and another in the barrel, the code will see that there are FTS hits in the event, and it will fill not only the trackcand of the fwd track, but it will create also a trackcand with the MVD+GEM hits of the barrel particle. Such second track is not a forward track. The opposite in the case of the PndSttMvdGemTrackingIdeal.

This means that maybe, once you run a barrel and a forward tracker together (both ideal, or one ideal and another real), you could have in the same event a track with MVD+STT+GEM hits, and another track with the same hits from MVD+GEM but w/o STT, since it is a "forward" track. The same track is reconstructed twice.

I hope I was clear. Maybe it would be better to put a selection, so that only TrackCand with at least one FTS or STT hit are stored. Maybe it would be good also to increase such number, since tracking with a single hit straw is impossible I believe.

Please let me know your opinions, maybe I read wrongly the code.

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by MartinJGaluska on Wed, 18 Sep 2013 21:37:43 GMT
View Forum Message <> Reply to Message

Hello Stefano,

I have just looked at PndFtsTrackerIdeal and I do believe that you are right in what you describe. I furthermore believe that it is a bug and I would like to suggest to only write out tracks which have at least 5 hits in the dipole field region of the FTS (as otherwise a

reconstruction of the particle's momentum is extremely hard). I will work on the modification for PndFtsTrackerIdeal. However, it might take me some time as I am at a PWA School until Friday next week with a tight program and I only have a netbook with me.

Best wishes,
Martin

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by MartinJGaluska on Thu, 19 Sep 2013 12:23:42 GMT
View Forum Message <> Reply to Message

Hello Stefano,

I wanted to start fixing the problem during today's lunch break, but I noticed that we might not have the described problem after all. (Sorry, I was quite tired when I looked through the code last night.) I looked at the following code excerpt from
void PndFtsTrackerIdeal::Exec(Option_t * option):

(Note that iDet == 0 corresponds to the FTS)

```
// Detector loop
  for(Int_t iDet=0;iDet<4;iDet++){
    if (kFALSE == fBranchActive[iDet]) continue; //skip manually switched off detector
    if(fVerbose>4) Info("Exec","Use detector %i",iDet);
    // Hit loop
    for (Int_t ih = 0; ih < fHits[iDet]->GetEntriesFast(); ih++) {
      ghit = (FairHit*) fHits[iDet]->At(ih);
      if(!ghit) {
        if(fVerbose>3) Error("Exec","Have no ghit %i, array size:
%i",ih,fHits[iDet]->GetEntriesFast());
        continue;
      }
      Int_t mchitid=ghit->GetRefIndex();
      if(mchitid<0) {
        if(fVerbose>3) Error("Exec","Have a negative mcHit %i",mchitid);
        continue;
      }
      myPoint = (FairMCPoint*)(fMCPoints[iDet]->At(mchitid));
      if(!myPoint) continue;
      Int_t trackID = myPoint->GetTrackID();
      if(trackID<0) continue;

      if(fVerbose>5) Info("Exec","Have a Hit %i at Track index %i",ih,trackID);

      // Continue Construction of a track candidate (start with FTS hits)
      // Track candidates and corresponding MC track index are saved in a map
      PndTrackCand* cand=candlist[trackID];
      if(NULL==cand){
        if(0!=iDet){
```

```
        if(fVerbose>5) Info("Exec","Skip Hit %i, it's not connected to a Track in FTS",ih);
        continue; // skip Tracks in MVD/GEM not going to FTS
      }
      if(fVerbose>5) Info("Exec","Create new PndTrack object %i",trackID);
      cand=new PndTrackCand();
      cand->setMcTrackId(trackID);
      if(fVerbose>5) Info("Exec","Creating new PndTrack object finished %i",trackID);
    }
    if(fVerbose>5) Info("Exec","add the hit %i to trackcand %i",ih,trackID);
    cand->AddHit(fBranchIDs[iDet],ih,myPoint->GetTime());
    // Figure out if the current hit is the earliest in the event
    if(!firstHit[trackID] || firstPoint[trackID]->GetTime() > myPoint->GetTime()) {
      firstHit[trackID]=ghit;
      firstPoint[trackID]=myPoint;
    }
    // or the latest one
    if(!lastHit[trackID] || lastPoint[trackID]->GetTime() < myPoint->GetTime()) {
      lastHit[trackID]=ghit;
      lastPoint[trackID]=myPoint;
    }

    candlist[trackID] = cand; // set
  }// end loop over hits
}// end loop over detectors
// now we have track candidates
```

The most important part is actually this one:

```
PndTrackCand* cand=candlist[trackID];
    if(NULL==cand){
      if(0!=iDet){
        if(fVerbose>5) Info("Exec","Skip Hit %i, it's not connected to a Track in FTS",ih);
        continue; // skip Tracks in MVD/GEM not going to FTS
      }
      if(fVerbose>5) Info("Exec","Create new PndTrack object %i",trackID);
      cand=new PndTrackCand();
      cand->setMcTrackId(trackID);
      if(fVerbose>5) Info("Exec","Creating new PndTrack object finished %i",trackID);
    }
```

So we first look at all FTS Hits individually and later all hits from other detectors in the event. For each hit we figure out which MC track caused them. ONLY if there is at least one FTS (iDet == 0) Hit from a MC track, we create a PndTrackCand. If there was no FTS hit for a MC track (iDet != 0), then the algorithm fails to load the corresponding PndTrackCand and it continues (meaning it skips the hit).

So, I believe that the PndFtsTrackerIdeal does exactly what you suggested (write out

PndTrackCands which have at least 1 FTS Hit). I would nevertheless like to change that behavior so that it requires at least 5 FTS hits in order to be more realisitic and if no one objects I will change the algorithm accordingly in the next few lunch breaks / nights.

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by StefanoSpataro on Thu, 19 Sep 2013 15:24:22 GMT
View Forum Message <> Reply to Message

I got the point, then it seems correct.
Maybe it would be good also to exclude, in low momentum particles, the tracks bent by the dipole and going forward again, since they create strange artifacts maybe not easy to fit, impossible to reconstruct I believe.

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by Tobias Stockmanns on Thu, 19 Sep 2013 15:37:52 GMT
View Forum Message <> Reply to Message

I still wonder why the FairLinks are not used for this purpose. They would allow to solve this problem in a much more elegant way and this task is exactly what they are build for.

Cheers,

Tobias

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by StefanoSpataro on Thu, 19 Sep 2013 15:43:52 GMT
View Forum Message <> Reply to Message

The task was written in 2011 by Ralf for the MVD tdr, when FairLinks were still fighting against tpc clusters and memory. It would be good an update of the task using the links, I agree.

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by MartinJGaluska on Thu, 19 Sep 2013 16:42:30 GMT
View Forum Message <> Reply to Message

Stefano Spataro wrote on Thu, 19 September 2013 17:24
Maybe it would be good also to exclude, in low momentum particles, the tracks bent by the dipole and going forward again, since they create strange artifacts maybe not easy to fit, impossible to reconstruct I believe.

Hello Stefano,

do you mean by that tracks which are bent in the dipole so much that they turn around and fly towards the barrel again (and therefore leave additional hits in the FTS)? If these events cause problems with the fitter, I will remove such tracks at the end of the loop on FTS hits. I plan to

check all (MC truth) time-ordered FTS hits associated to a given PndTrackCand and check if the z-component is increasing. If not, I will remove the PndTrackCand.

Tobias Stockmanns wrote on Thu, 19 September 2013 17:37I still wonder why the FairLinks are not used for this purpose. They would allow to solve this problem in a much more elegant way and this task is exactly what they are build for.

Hello Tobias,

I will try to move the implementation of PndFtsTrackerIdeal to FairLinks. Can you recommend a task which I can use as an example? Is it a good idea to start looking at PndTrackingQualityAnalysis?

Kind regards,
Martin

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by StefanoSpataro on Thu, 19 Sep 2013 17:25:34 GMT
View Forum Message <> Reply to Message

Martin J Galuska wrote on Thu, 19 September 2013 18:42 Hello Stefano,

do you mean by that tracks which are bent in the dipole so much that they turn around and fly towards the barrel again (and therefore leave additional hits in the FTS)? If these events cause problems with the fitter, I will remove such tracks at the end of the loop on FTS hits. I plan to check all (MC truth) time-ordered FTS hits associated to a given PndTrackCand and check if the z-component is increasing. If not, I will remove the PndTrackCand.

Exactly. But also inside the dipole, they could hit the same layer twice, once in the forward direction and once in the backward direction. Maybe some algorithm could be able to find circles in the dipole, but I believe it will be quite tough to reconstruct the backward segment in FTS1 and 2. Of course this Moreover, sometimes you have secondaries starting inside the diopole region, with the first hit in fts 34. Such tracks will be impossible tor econstruct. Then maybe one should ask to have at least a minimum number of fired FTS12 straws, and a minimum for FTS34.
Let's put in this way:i believe we should clean a bit the ideal track sample, to select only tracks that the real algorithm will be able to reconstruct.If we leave them, our efficiency will be spoiled to a too high value compared to real case.

My humble opinion, comments and suggestions are welcome.

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by MartinJGaluska on Fri, 20 Sep 2013 16:51:36 GMT
View Forum Message <> Reply to Message

Hello Stefano and other collaborators,

I have just uploaded an updated version of PndFtsTrackerIdeal which is supposed to get rid off tracks that turn around in the forward dipole field and it also should not write find any tracks which have less than 5 hits in the FTS.

I write "should" because I currently do not have time to check it thoroughly, but the code compiles. I hope someone will want to run some simulation and check if the desired behavior is achieved.

I am also open to suggestions for improvements and I will certainly work on the rewrite using FairLinks.

Kind regards,
Martin

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by MartinJGaluska on Mon, 24 Feb 2014 15:18:16 GMT
View Forum Message <> Reply to Message

Dear all,

after having changed the standard behavior of the FTS ideal tracking to make it behave more realistically from the FTS tracking point of view, Donghee noticed a problem in the tracking

I required a track to be found by the FTS tracking to have at least 5 FTS hits. Previously, the FTS ideal tracker "found" all tracks that had at least 1 hit in the FTS.

As I said, from the FTS tracking point of view that behavior is more realisitic. However, currently there is only a tracking starting from STT + MVD and from FTS in the code. Both tracking algorithms find mostly distinct sets of tracks so that a merge is easily done. Hits from GEM are only added to tracks found by FTS and by STT + MVD, but there is no tracking starting from GEM being used in the current version of the code.

As a workaround I implemented PndFtsTrackerIdeal::SetMinFtsHitsPerTrack(int); to set the number back to 1 to have an overall detector performance that is similar to before the changes in the FTS ideal tracking. I have just changed the default value to 1 to avoid possible problems and confusion, especially when the simulation campaigns will be executed and new results will be compared with old ones.

At this point I suggest to use the value of 5 for standalone performance studies of the FTS only.

Here is how:

```
PndFtsTrackerIdeal* trackFts = new PndFtsTrackerIdeal();
  trackFts->SetMinFtsHitsPerTrack(1);
  trackFts->SetRelativeMomentumSmearing(0.05);
  trackFts->SetVertexSmearing(0.05, 0.05, 0.05);
  trackFts->SetTrackingEfficiency(1.);
  trackFts->SetTrackOutput("FtsIdealTrack");
```

```
trackFts->SetPersistence(kFALSE);
fRun->AddTask(trackFts);
```

Kind regards,
Martin

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by StefanoSpataro on Wed, 05 Mar 2014 15:14:46 GMT
View Forum Message <> Reply to Message

Hi,
I think the value 5 is the more correct, since this is tracking for FTS detector.
Tracking with only MVD+GEM is not present in the default reconstruction, but it is not a task of that track finder.

Subject: Re: Bug in PndFts/SttMvdGemTrackingIdeal ?
Posted by MartinJGaluska on Fri, 07 Mar 2014 16:35:07 GMT
View Forum Message <> Reply to Message

After a discussion with Stefano about this issue, the decision was made that the standard behavior should be to require a minimum of 5 FTS hits.

If necessary, users can change this number as explained in an earlier post.
