Quality Assurance for tracking

Lia Lavezzi, Stefano Spataro
University of Torino & INFN

Tobias Stockmanns , André Zambanini
Juelich Forschungszentrum

LIII PANDA Collaboration Meeting – Computing Session
Uppsala, 8 – 12 June 2015

fusion of the codes

AGREED @ last collaboration meeting in Giessen

- **Ideal track finding:**
  - start/end position/momentum from the MC point
  - sorting via the *time of flight*
- **QA information:**
  - store a *MCTrackInfo* object for each reconstructable MC track
  - store a *RecoTrackInfo* object for each reco track

- **Ideal track finding:**
  - Perform it via the new *FairLink* system
- **QA information:**
  - Existing functors for selection criteria
  - Structure which uses the FairLinks to evaluate a track

The Data objs

PndTrk**MCTrackInfo**

- **# MC points** in each detector*
- index of the associated **PndMCTrack**
- array of indices of associated **PndTrack**s
- **MC position/momentum** @1st /last points
- **MC charge**
- **reconstructability** flag

PndTrk**RecoTrackInfo**

- **# true** hits in each detector*
- **# fake** hits in each detector*
- **# missing** hits in each detector*
- **PndTrkMCTrackInfo** object
- index of the associated **PndMCTrack**
- index of the associated **PndTrack**
- **reco position/momentum** @1st/last points
- **reco charge**
- **true/clone** flag

** mvd pixel, mvd strip, stt, gem*

Recall the procedure

"NEW LINKS" IDEAL TRACK FINDER

- Finds all the tracks which are *reconstructable*
- Fills a TCA of Ideal PndTrack(Cand)s
- Fills the PndTrackingQualityMCInfo object TCA *new*

one mc info for each mc track

QA TASK

- Analyzes the single reconstructed PndTrack(Cand)
- Compares it to the Ideal PndTrack(Cand)
- Fills the PndTrackingQualityRecoInfo object TCA *new*
- Decides which track is *true* and which are *clones*

one reco info for each reco track

The full procedure now runs on MVD pixel/strip, STT, GEM hits (FTS?)

Testing

- Apply the QA procedure to the ideally found tracks with the OLD ideal track finder, the "good old" **PndSttMvdGemTrackingIdeal**

Reco *ideal*, in this case

```
PndSttMvdGemTrackingIdeal* trackStt = new PndSttMvdGemTrackingIdeal();
trackStt->SetRelativeMomentumSmearing(0.);
trackStt->SetVertexSmearing(0., 0., 0.);
trackStt->SetTrackingEfficiency(1.);
trackStt->SetTrackOutput("SttMvdGemIdealTrack");
fRun->AddTask(trackStt);
```

Testing

- Apply the QA procedure to the ideally found tracks with the OLD ideal track finder, the "good old" **PndSttMvdGemTrackingIdeal**

Reco *ideal*, in this case

```
PndSttMvdGemTrackingIdeal* trackStt = new PndSttMvdGemTrackingIdeal();
trackStt->SetRelativeMomentumSmearing(0.);
trackStt->SetVertexSmearing(0., 0., 0.);
trackStt->SetTrackingEfficiency(1.);
trackStt->SetTrackOutput("SttMvdGemIdealTrack");
fRun->AddTask(trackStt);
```

QA evaluation

```
PndMCIdealTrackFinderNewLinks* idealTracking = new PndMCIdealTrackFinderNewLinks();
idealTracking->AddBranchName("MVDHitsPixel");
idealTracking->AddBranchName("MVDHitsStrip");
idealTracking->AddBranchName("STTHit");
idealTracking->AddBranchName("GEMHit");
fRun->AddTask(idealTracking);
```

Testing

- Apply the QA procedure to the ideally found tracks with the OLD ideal track finder, the "good old" **PndSttMvdGemTrackingIdeal**

Reco *ideal*, in this case

```
PndSttMvdGemTrackingIdeal* trackStt = new PndSttMvdGemTrackingIdeal();
trackStt->SetRelativeMomentumSmearing(0.);
trackStt->SetVertexSmearing(0., 0., 0.);
trackStt->SetTrackingEfficiency(1.);
trackStt->SetTrackOutput("SttMvdGemIdealTrack");
fRun->AddTask(trackStt);
```

QA evaluation

```
PndMCIdealTrackFinderNewLinks* idealTracking = new PndMCIdealTrackFinderNewLinks();
idealTracking->AddBranchName("MVDHitsPixel");
idealTracking->AddBranchName("MVDHitsStrip");
idealTracking->AddBranchName("STTHit");
idealTracking->AddBranchName("GEMHit");
fRun->AddTask(idealTracking);
```

```
PndTrackingQualityTaskNewLinks* trackingQA = new
PndTrackingQualityTaskNewLinks("SttMvdGemIdealTrack", "IdealTrack");
fRun->AddTask(trackingQA);
```

Testing

- Apply the QA procedure to the ideally found tracks with the OLD ideal track finder, the "good old" **PndSttMvdGemTrackingIdeal**

Reco *ideal*, in this case

```
PndSttMvdGemTrackingIdeal* trackStt = new PndSttMvdGemTrackingIdeal();
trackStt->SetRelativeMomentumSmearing(0.);
trackStt->SetVertexSmearing(0., 0., 0.);
trackStt->SetTrackingEfficiency(1.);
trackStt->SetTrackOutput("SttMvdGemIdealTrack");
fRun->AddTask(trackStt);
```

QA evaluation

```
PndMCIdealTrackFinderNewLinks* idealTracking = new PndMCIdealTrackFinderNewLinks();
idealTracking->AddBranchName("MVDHitsPixel");
idealTracking->AddBranchName("MVDHitsStrip");
idealTracking->AddBranchName("STTHit");
idealTracking->AddBranchName("GEMHit");
fRun->AddTask(idealTracking);
```

```
PndTrackingQualityTaskNewLinks* trackingQA = new
PndTrackingQualityTaskNewLinks("SttMvdGemIdealTrack", "IdealTrack");
fRun->AddTask(trackingQA);
```

output of the PR under evaluation

output of the ideal track finder

Testing

- Apply the QA procedure to the ideally found tracks with the OLD ideal track finder, the "good old" **PndSttMvdGemTrackingIdeal**

Reco *ideal*, in this case

```
PndSttMvdGemTrackingIdeal* trackStt = new PndSttMvdGemTrackingIdeal();
```

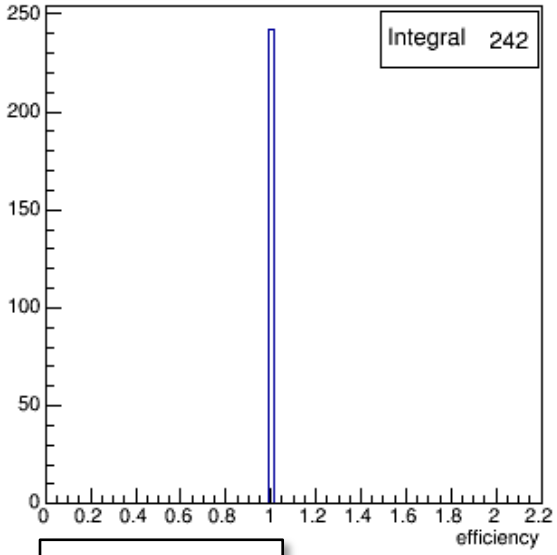
- The PR under evaluation, in this case:
  - must find all the hits in the track → efficiency must be 1
  - must find only the true hits in the track → purity must be 1
  - must have the correct start/end position/momentum → residuals must be 0

```
idealTracking->AddBranchName("MVDHitsPixel");
idealTracking->AddBranchName("MVDHitsStrip");
idealTracking->AddBranchName("STTHit");
idealTracking->AddBranchName("GEMHit");
fRun->AddTask(idealTracking);
```

```
PndTrackingQualityTaskNewLinks* trackingQA = new
PndTrackingQualityTaskNewLinks("SttMvdGemIdealTrack", "IdealTrack");
fRun->AddTask(trackingQA);
```

Results: efficiency

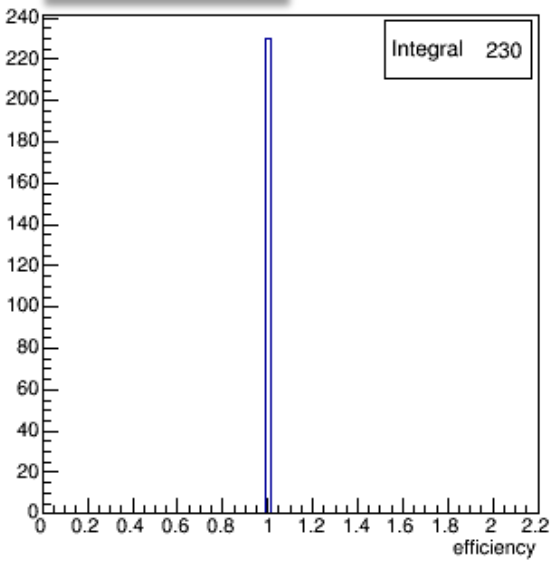
MVD pixel



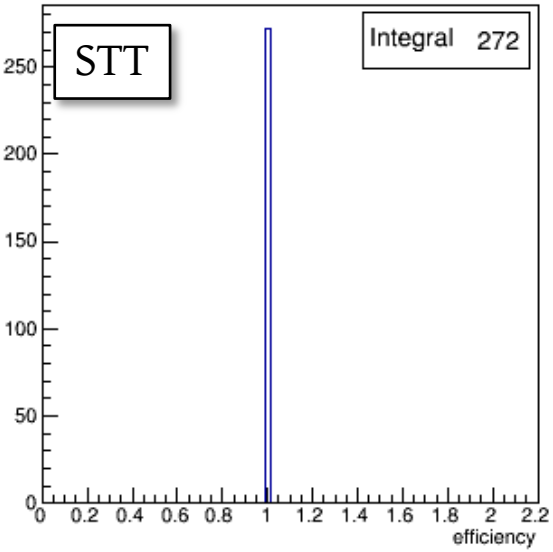
Simulation:	Reconstruction:
100 events	ideal
3 μ^-/event	no Kalman filter applied
1.5 GeV/c	
$\theta \in [0.1, 110]°$	

```
cbmsim.Draw(''RecoTrackInfo.GetMvdPixelEfficiency()'',
   ''RecoTrackInfo.GetMvdPixelEfficiency() != -1'')
```

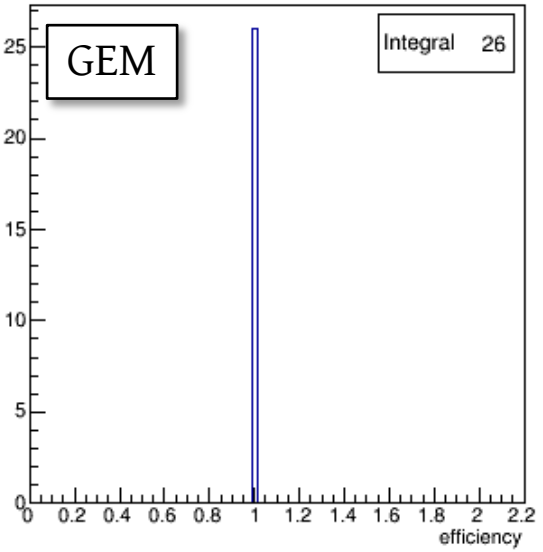
MVD strip



STT

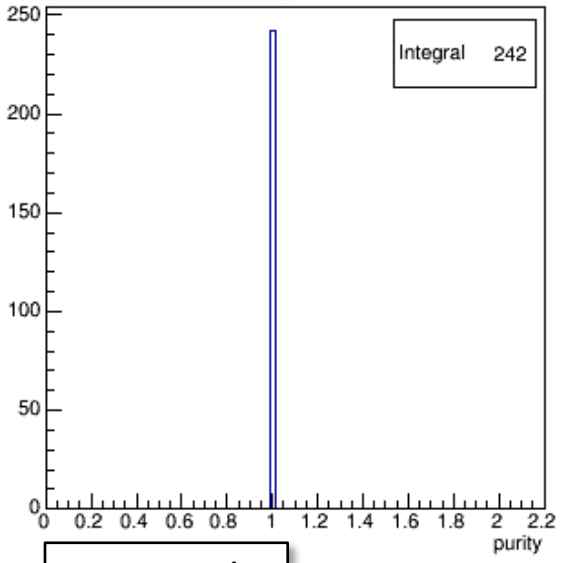


GEM



Results: purity

MVD pixel

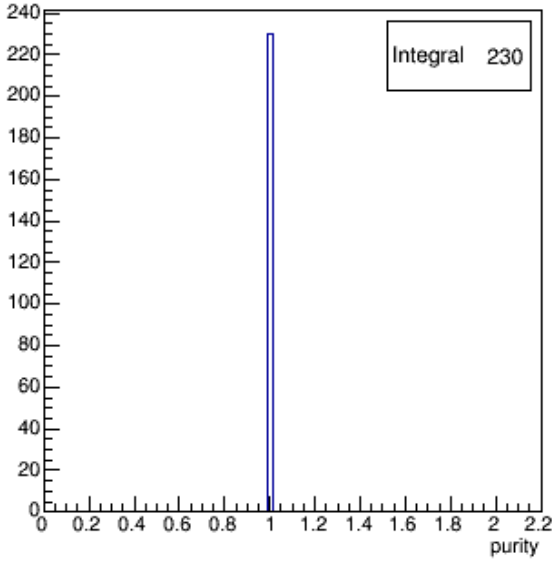


Simulation:
100 events
3 μ^-/event
1.5 GeV/c
$\theta \in [0.1, 110]°$

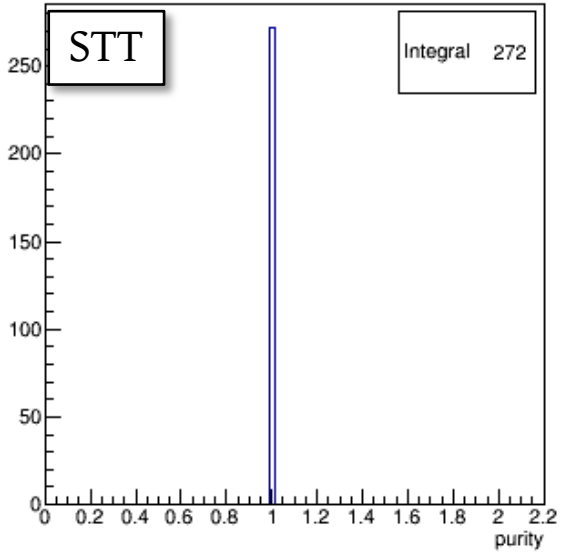
Reconstruction:
ideal
no Kalman filter applied

```
cbmsim.Draw(''RecoTrackInfo.GetMvdPixelPurity()'',
   ''RecoTrackInfo.GetMvdPixelPurity() != -1'')
```

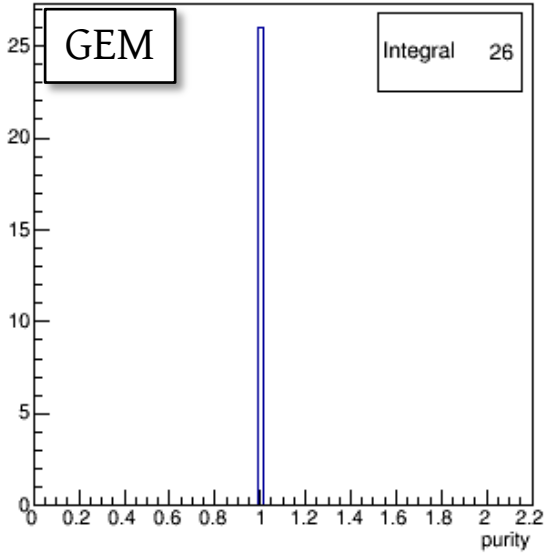
MVD strip



STT

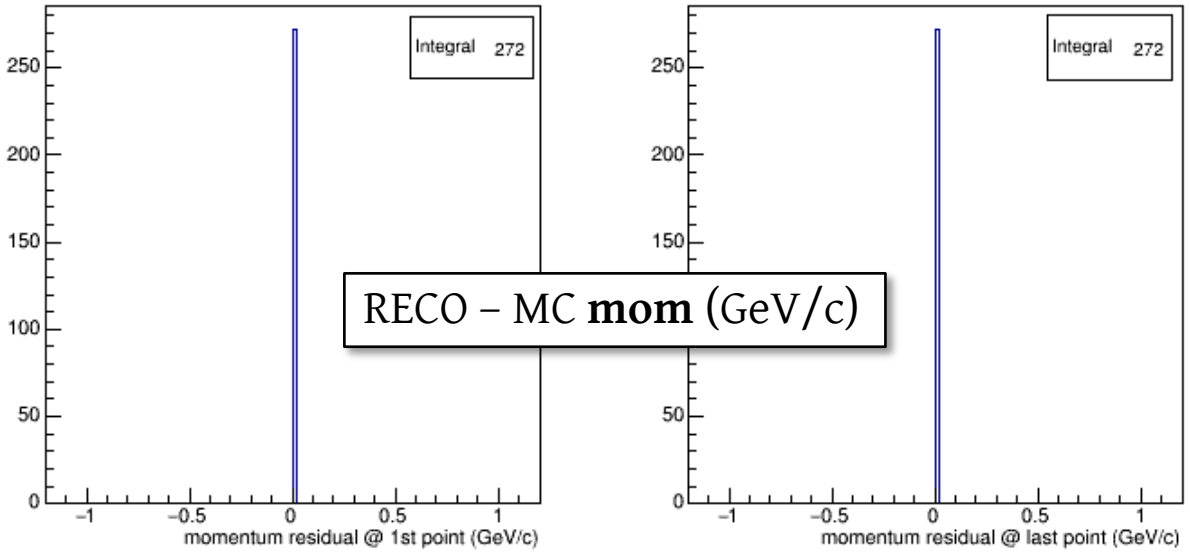


GEM

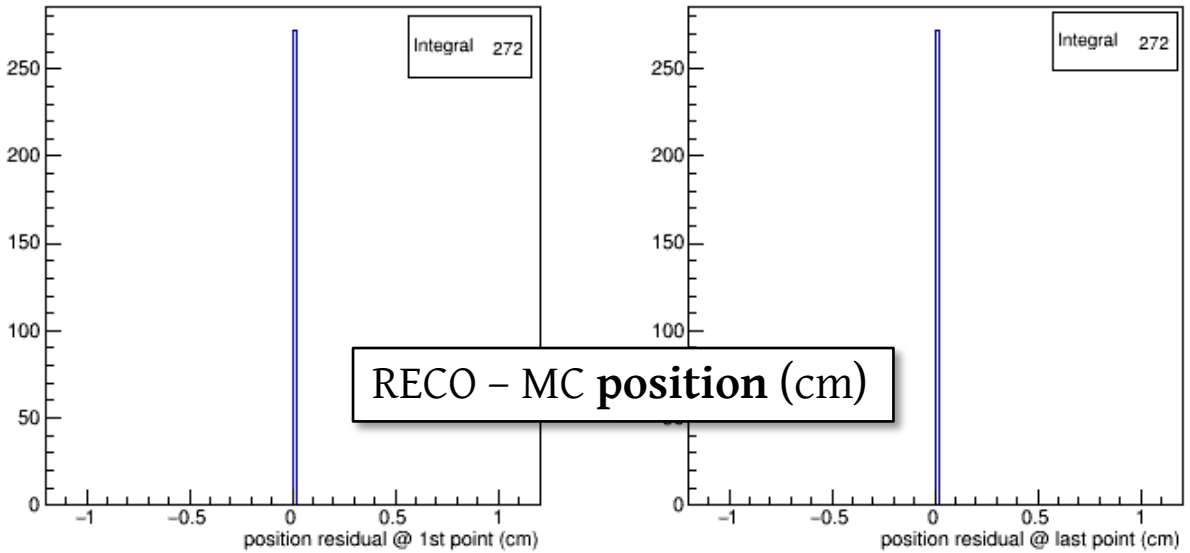


Results: residuals

```
cbmsim.Draw(``RecoTrackInfo.GetMomentumFirst().Mag() -
RecoTrackInfo.GetMCTrackInfo().GetMomentumFirst().Mag()'')
```



RECO – MC **mom** (GeV/c)



RECO – MC **position** (cm)

Simulation:	Reconstruction:
100 events	ideal
3 μ^-/event	no Kalman filter applied
1.5 GeV/c	
$\theta \in [0.1, 110]°$	

A realistic example

Simulation:	Reconstruction:
100 events	real, i.e. Standard PR +
3 μ^-/event	GEM extension
1.5 GeV/c	no Kalman filter applied
$\theta \in [0.1, 110]°$	

A realistic example

Simulation:	Reconstruction:
100 events	real, i.e. Standard PR +
3 μ^-/event	GEM extension
1.5 GeV/c	no Kalman filter applied
$\theta \in [0.1, 110]°$	

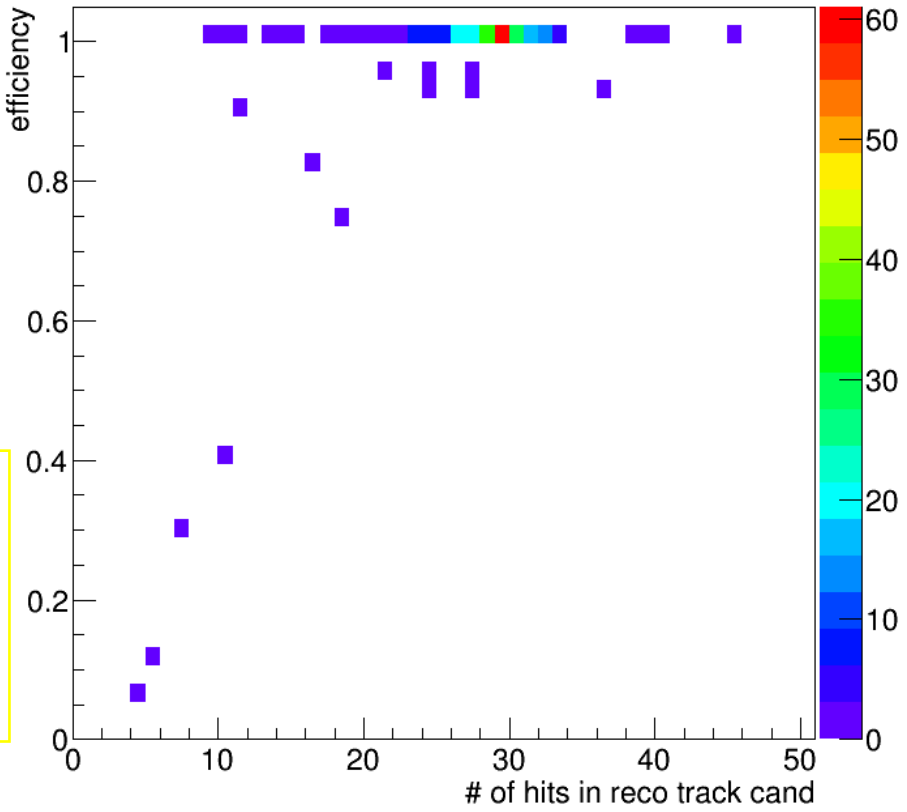
```
... Standard macro/run/reco_complete.C
```

```
PndMCIdealTrackFinderNewLinks* idealTracking = new PndMCIdealTrackFinderNewLinks();
idealTracking->AddBranchName("MVDHitsPixel");
idealTracking->AddBranchName("MVDHitsStrip");
idealTracking->AddBranchName("STTHit");
idealTracking->AddBranchName("GEMHit");
fRun->AddTask(idealTracking);
```

```
PndTrackingQualityTaskNewLinks* trackingQA = new
PndTrackingQualityTaskNewLinks("SttMvdGemTrack", "IdealTrack");
fRun->AddTask(trackingQA);
```

A realistic example

Simulation:	Reconstruction:
100 events	real, i.e. Standard PR +
3 μ^-/event	GEM extension
1.5 GeV/c	no Kalman filter applied
$\theta \in [0.1, 110]°$	



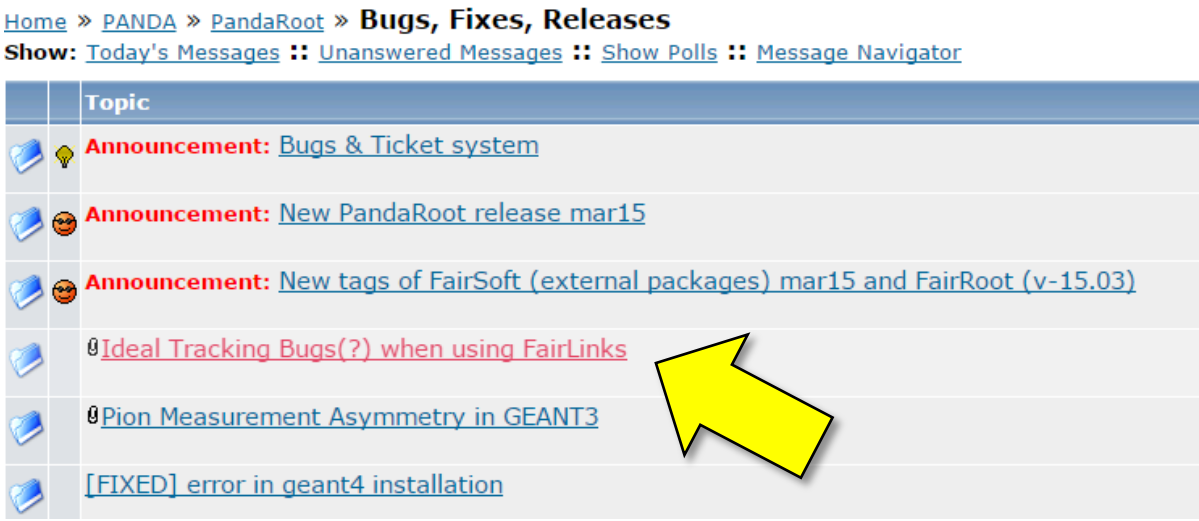

```
TFile filesim(''tst.root'');
TTree *cbmsim = (TTree*) filesim.Get("cbmsim");
cbmsim->AddFriend("cbmsim", "reco_complete.root");

cbmsim.Draw("RecoTrackInfo.GetEfficiency():SttMvdGem
TrackCand[RecoTrackInfo.GetRecoTrackID()].GetNHits()
", "", "colz")
```

''RecoTrackInfo.**GetEfficiency**():SttMvdGemTrackCand[RecoTrackInfo.GetRecoTrackID()].GetNHits()''

CUT: STT hits ≤ 25

- André found a bug: the first and last point were not correctly set in the *old* ideal track finder: PndSttMvdGemTrackingIdeal

- The cause was the cut on the maximum number of STT hits associated to the track



- Some time ago, a limit of 25 STT hits was set, since for more hits there were problems in the Kalman reconstruction

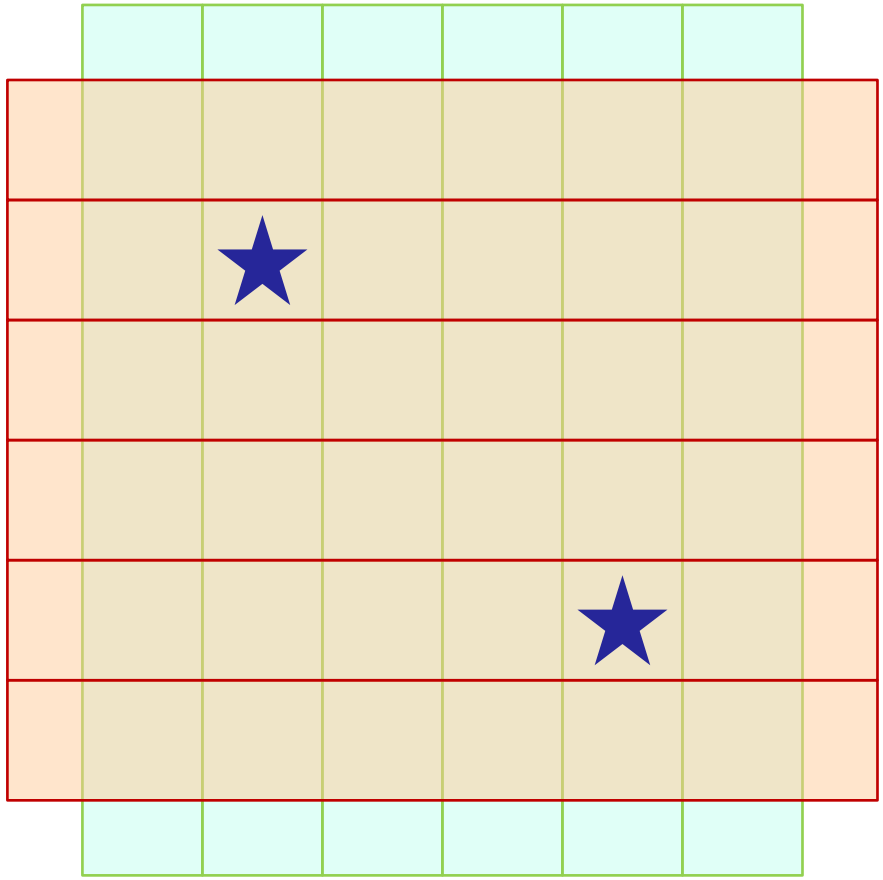
...BUT...

- The cut was by mistake set on the number of *total* associated STT hits, instead of the number of STT hits on a *single track* → the problem showed up

- Now **the cut has been removed** and the first/last point result correctly defined

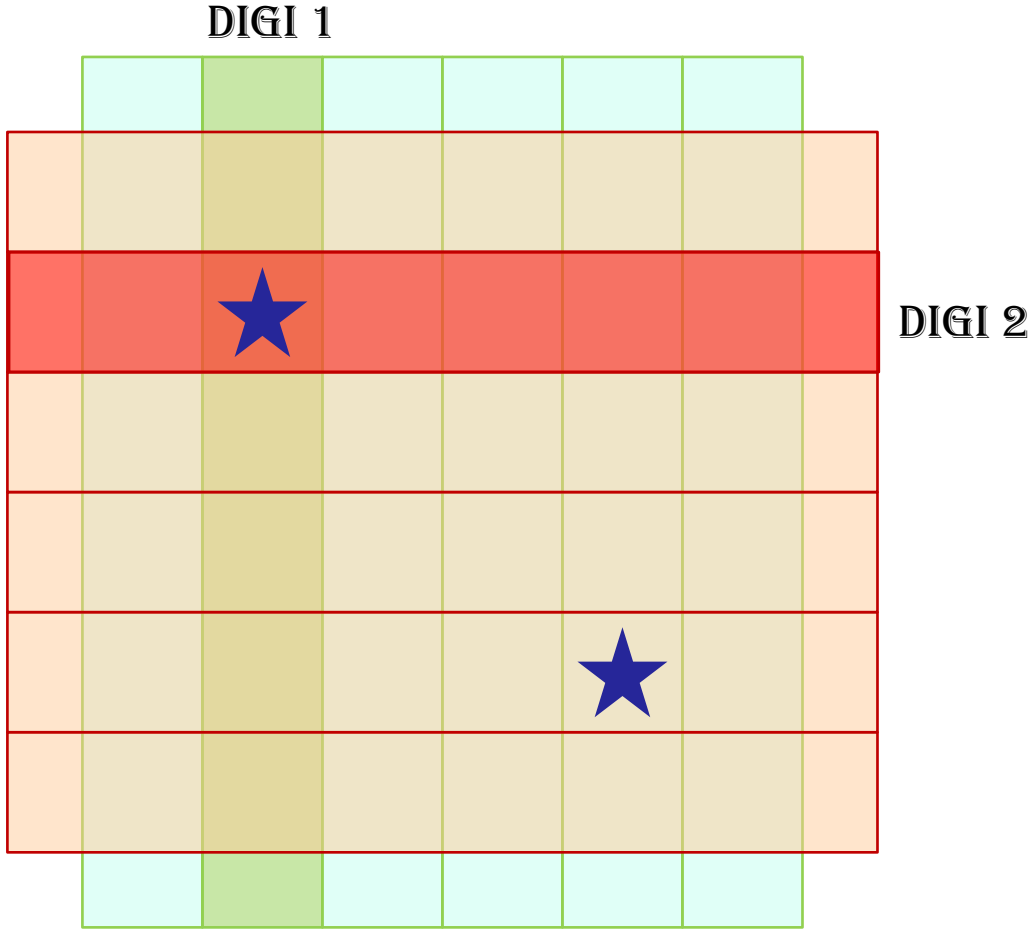
The combinatorial issue

- Now for MVD strip and GEM, due to the presence of the combinatorial hits, only the hits connected to one MC point are considered as true hits, the other are considered as combinatorial background hits



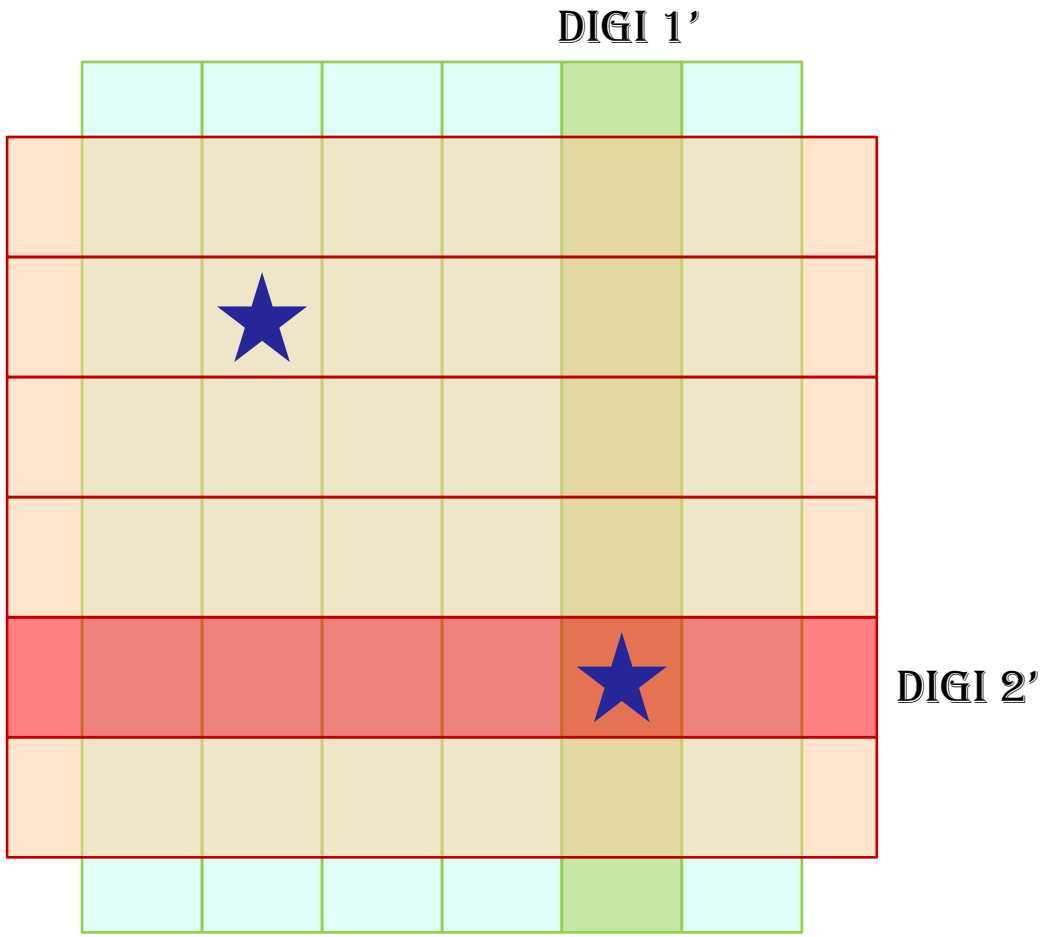
The combinatorial issue

- Now for MVD strip and GEM, due to the presence of the combinatorial hits, only the hits connected to one MC point are considered as true hits, the other are considered as combinatorial background hits



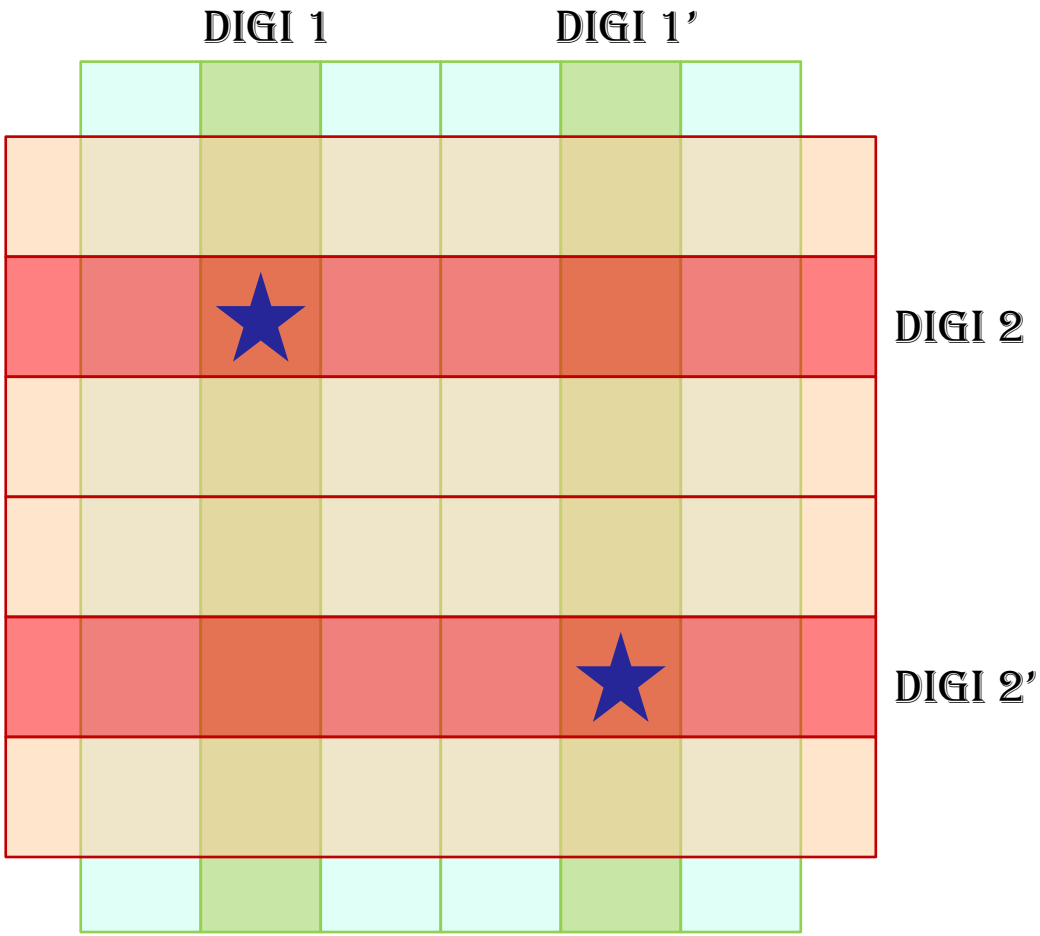
The combinatorial issue

- Now for MVD strip and GEM, due to the presence of the combinatorial hits, only the hits connected to one MC point are considered as true hits, the other are considered as combinatorial background hits



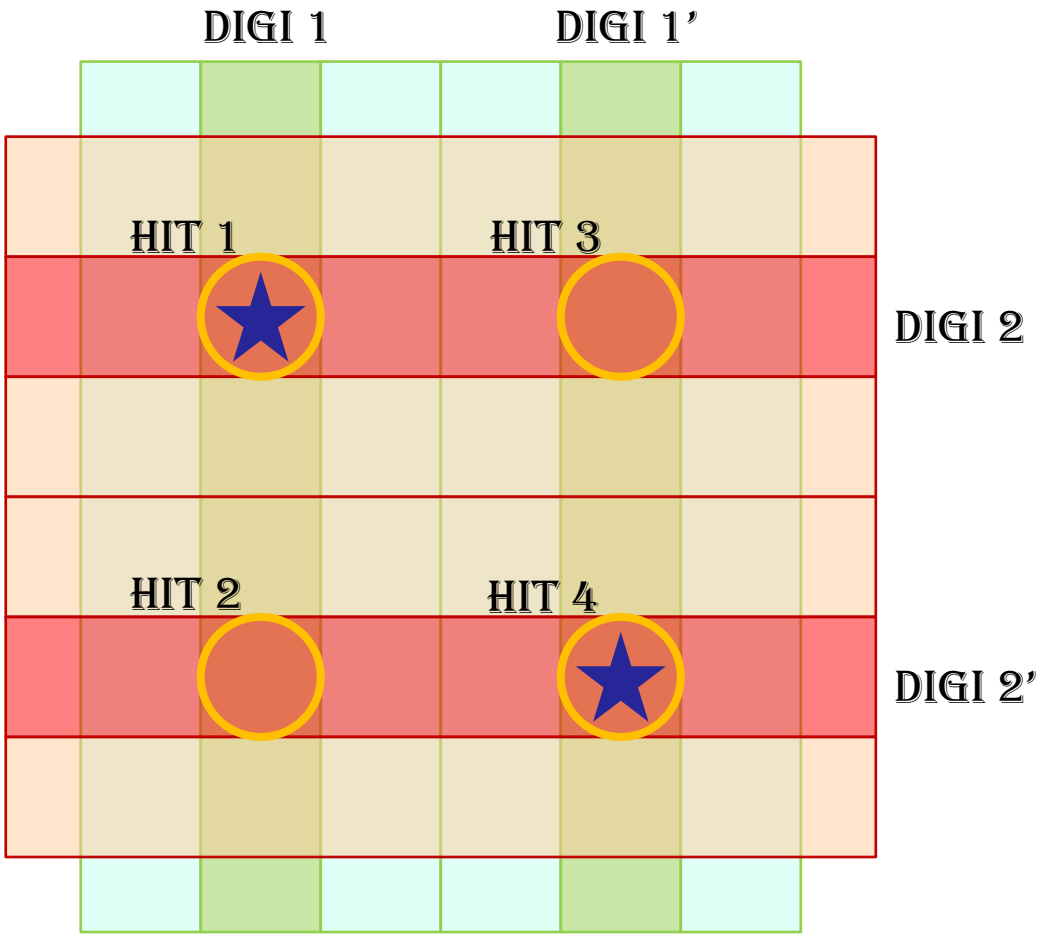
The combinatorial issue

- Now for MVD strip and GEM, due to the presence of the combinatorial hits, only the hits connected to one MC point are considered as true hits, the other are considered as combinatorial background hits



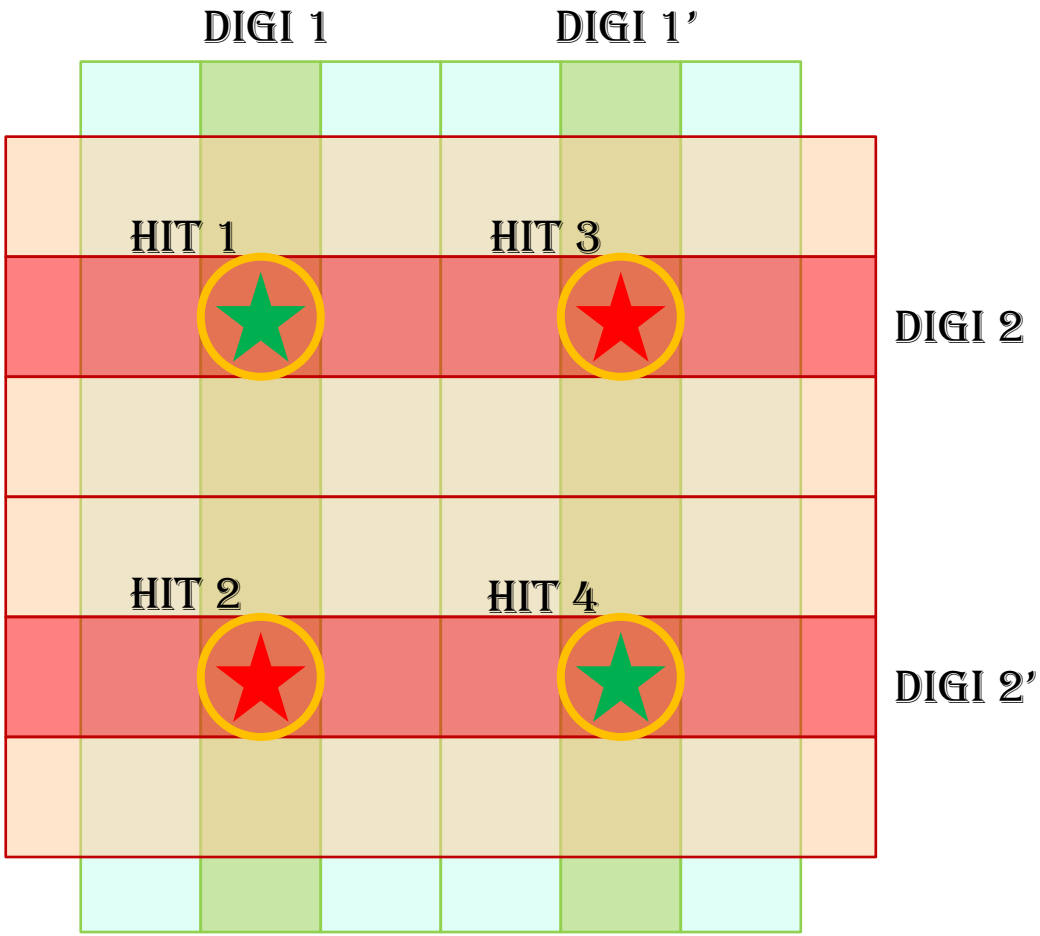
The combinatorial issue

- Now for MVD strip and GEM, due to the presence of the combinatorial hits, only the hits connected to one MC point are considered as true hits, the other are considered as combinatorial background hits



The combinatorial issue

- Now for MVD strip and GEM, due to the presence of the combinatorial hits, only the hits connected to one MC point are considered as true hits, the other are considered as combinatorial background hits



Where you can find it

source: pandaroot / trunk / PndMCMatchNewLinks @ 27775

Name
../
CMakeLists.txt
PndMCIdealTrackFinderNewLinks.cxx
PndMCIdealTrackFinderNewLinks.h
PndMCMatchNewLinksLinkDef.h

Everything is in the **trunk/** branch

source: pandaroot / trunk / PndTools / TrackingQA @ 27775

Name
../
CMakeLists.txt
dbuilder.C
onlineDBuilderLinkDef.h
OnlineDBuilderTask.cxx
OnlineDBuilderTask.h
PndTrackingQualityAnalysis.cxx
PndTrackingQualityAnalysis.h
PndTrackingQualityAnalysisNewLinks.cxx
PndTrackingQualityAnalysisNewLinks.h
PndTrackingQualityData.cxx
PndTrackingQualityData.h
PndTrackingQualityMCInfo.cxx
PndTrackingQualityMCInfo.h
PndTrackingQualityRecoInfo.cxx
PndTrackingQualityRecoInfo.h
PndTrackingQualityTask.cxx
PndTrackingQualityTask.h
PndTrackingQualityTaskNewLinks.cxx
PndTrackingQualityTaskNewLinks.h
trackingQA.C
trackingQANewLinks.C
TrackingQualityAnalysis_LinkDef.h

Still missing

- Filling of the reconstructability flag in the MCTrackInfo object, via the functors
- Filling of the isClone/isTrue flag
- STT parallel and skewed separation for efficiency, purity, etc.
- Task to fill the histograms for QA
  - 7 histos: Global efficiency, for all the primary tracks, for all the detectors
    Efficiency for MVD pixel, MVD strips, Stt parallel, Stt skewed, gem, FTS (skewed? parallel?)
  - 7 histos: Global Purity, for all the primary tracks, for all the detectors
  - 2 histos:
    Number of reco tracks correlated to MC track, for primaries (>1 → clones)
    Number of MC tracks correlated to reco tracks for primaries (>1 → broken tracks)
  - 7 histos: Resolution at the first point:
    delta_p, delta_pz, delta_pperp, delta_theta, delta_phi, delta_position, delta_charge
  - 7 histos: Resolution at the last point:

The End