ITS tracking:
- factorization of the features
- material from TGeo

A.Dainese (INFN Legnaro)

Motivation

- AliITStrackerMI (default Kalman filter in ITS) was optimized by Marian for efficiency and speed in high-mult. events
- Features
  - required number of clusters/track
  - cluster sharing between two tracks
  - usage of amplitude info in SDD and SSD
  - V0 finding on-the-fly during tracking
  - ...

  and “numbers”
  - cuts (definition of primary track, track-cluster chi2, ...)
  - geometry & material

  are mostly hardcoded

→ **change the features into *options***

→ **extract the numbers from the code:**

→ **cuts have to be easily adjustable (tuning)**

→ **geometry & material should be managed by TGeo**

Work plan

- New class AliITSRecoParam — *"DONE"*
  - steering class to manage options of ITS tracking
  - each cut in AliITStrackerMI becomes a data member of this class
  - other numbers used in tracking (e.g. # ITS layers, beam pipe and shields radii, thicknesses, X0) defined as static in AliITSRecoParam.h
    - replaces AliITSrecoV2.h (was a container for static variables)
  - constantly growing...
- Get geometry and material budget from TGeo — *"almost DONE"*
  - material of "dead zones" (between TPC and ITS, thermal shields between SPD and SDD & between SDD and SSD, beam pipe)
  - material of each layer
  - position of gaps in z between SPD modules
- New class AliITSClusterParam — *STARTED*
  - deals with cluster shape (error) parametrization (currently done in AliITStrackerMI::GetError, GetNTeor)
  - also methods to extract parameters from real data

Class AliITSRecoParam

usage

- AliITSReconstructor has a new data member:
  static AliITSRecoParam *fgkRecoParam
- Tracking options set from the rec.C macro:

```
AliITSRecoParam * itsRecoParam = AliITSRecoParam::GetLowFluxParam();
 //AliITSRecoParam * itsRecoParam = AliITSRecoParam::GetHighFluxParam();
 //AliITSRecoParam * itsRecoParam = AliITSRecoParam::GetCosmicTestParam();
itsRecoParam->SetUseTGeoInTracker(1);
itsRecoParam->SetFindV0s(kTRUE);
itsRecoParam->SetAllowSharedClusters(kTRUE);
itsRecoParam->SetUseAmplitudeInfo(kTRUE);
AliITSReconstructor::SetRecoParam(itsRecoParam);
```

- From the tracker(s), numbers retrieved with:
  AliITSReconstructor::GetRecoParam()->GetMinPtForProlongation()

recent update (not yet on CVS)

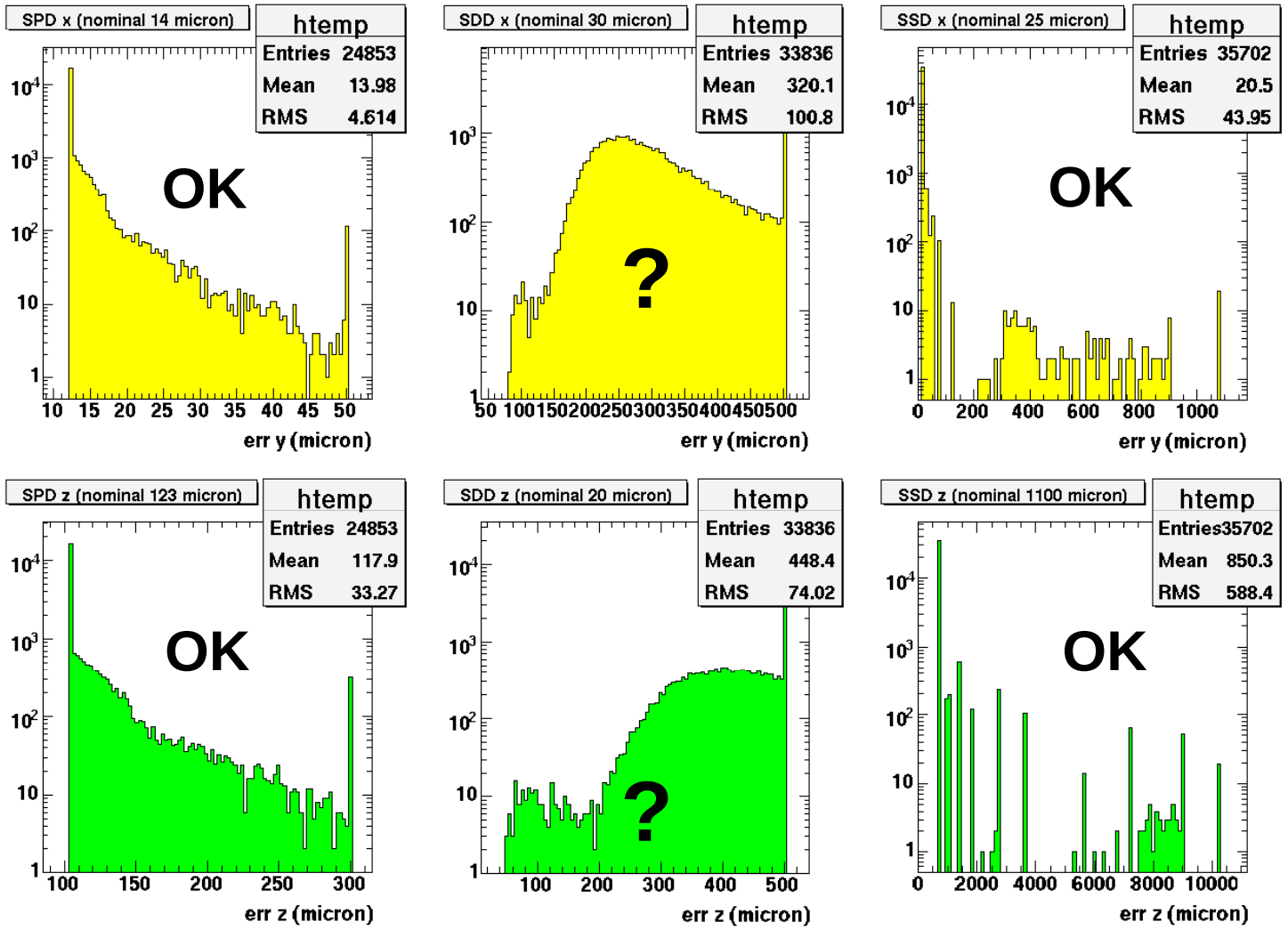
- Possibility to switch-off the use of amplitude info (dE signal) in TPC and in SSD/SDD as criterion for cluster-track matching

Class AliITSClusterParam

- Class has been created
- At the moment, just moved there two methods from AliITStrackerMI:
  - GetNTeor (expected cluster size)
  - GetError (expected cluster errors, used to calculate cluster-track χ^2)
- Will contain also methods to extract from MC/data size and error parametrizations
- Looking into GetError brought to interesting “discoveries”...

Issue #1: SDD cluster errors

- Errors from AliITStrackerMI::GetError

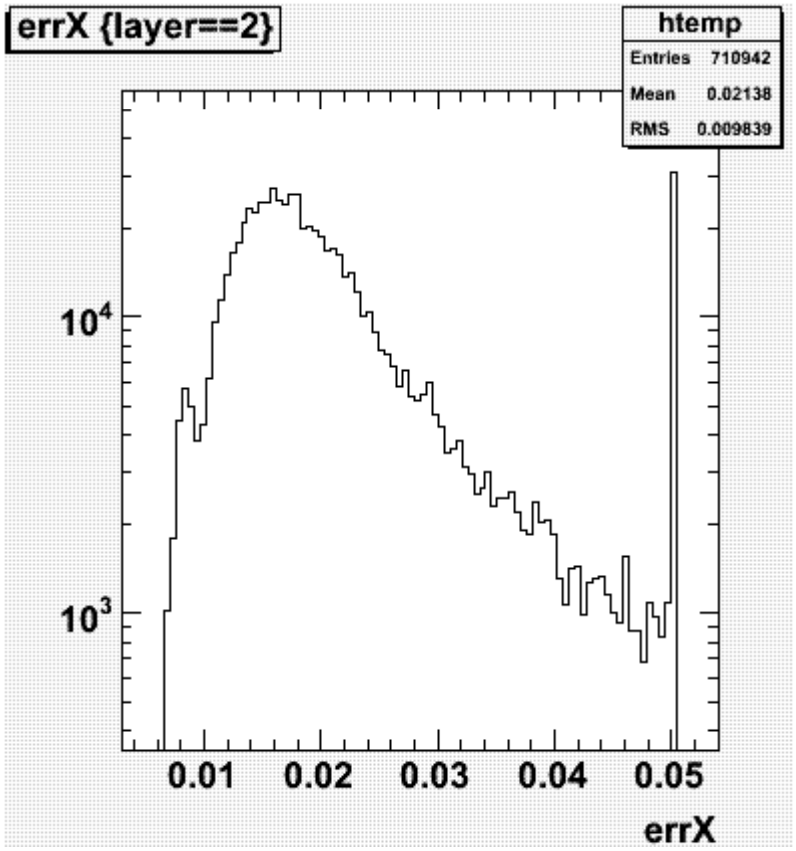


Issue #1: SDD cluster errors

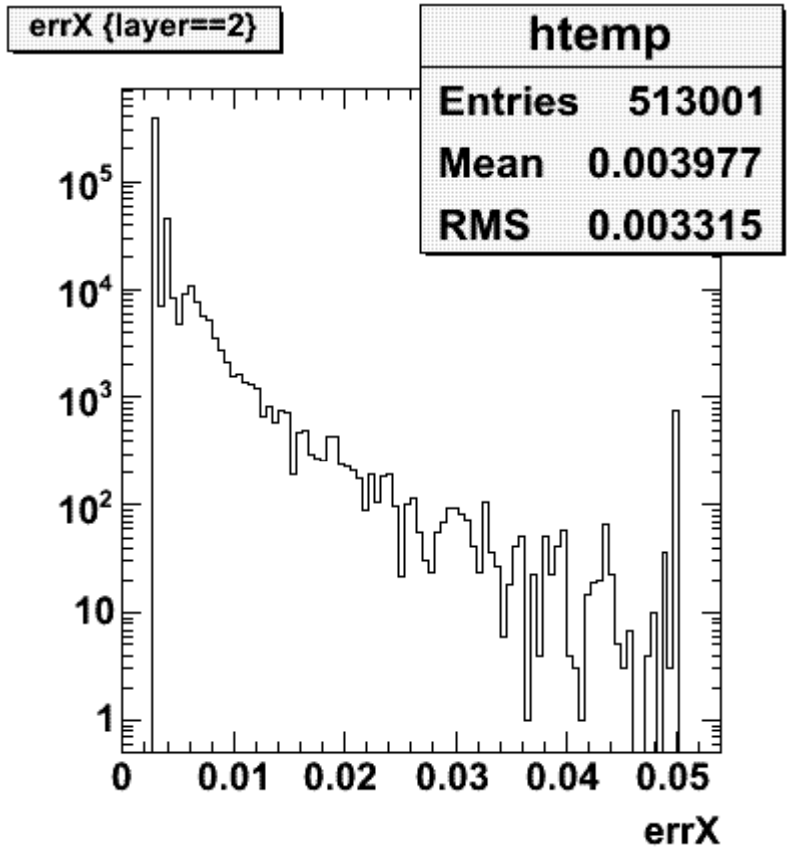
- Problem investigated and solved by F.Prino
- Origin(s) of the problem:
  - SDD charge converted from ADC to keV on April 3rd 2007 without modifying the corresponding cut values (hard coded) in AliITStrackerMI
    - Charge in SDD clusters no longer matching TPC track charge ⇒ SDD cluster errors enlarged (by a factor ≈ 3) by the tracker
  - Bug in the calculation of the cluster size along anodes introduced when removing hard coded numbers in AliITSClusterFinderV2SDD (June 1st 2007)
    - SDD cluster size very large (always = 5 anodes) ⇒ SDD cluster errors enlarged (angain by a factor ≈ 3) by the tracker
- Problems now solved (AliITSClusterFinderV2SDD, AliITStrackerMI)

Issue #1: SDD cluster errors

- Distribution of errors on SDD clusters given by AliITStrackeMI::GetError()



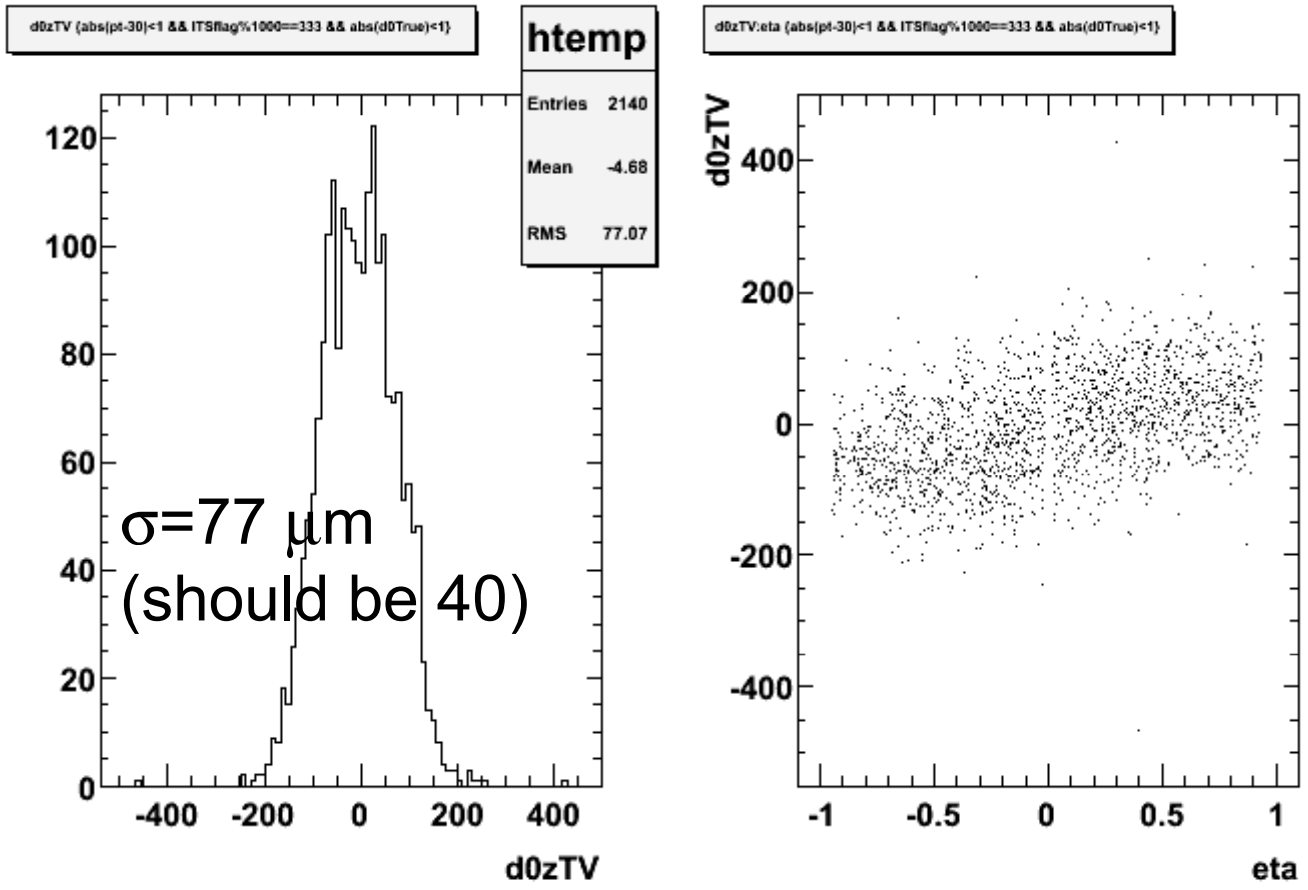
AliRoot HEAD August 2nd



AliRoot HEAD September 27th

Issue #1: SDD cluster errors

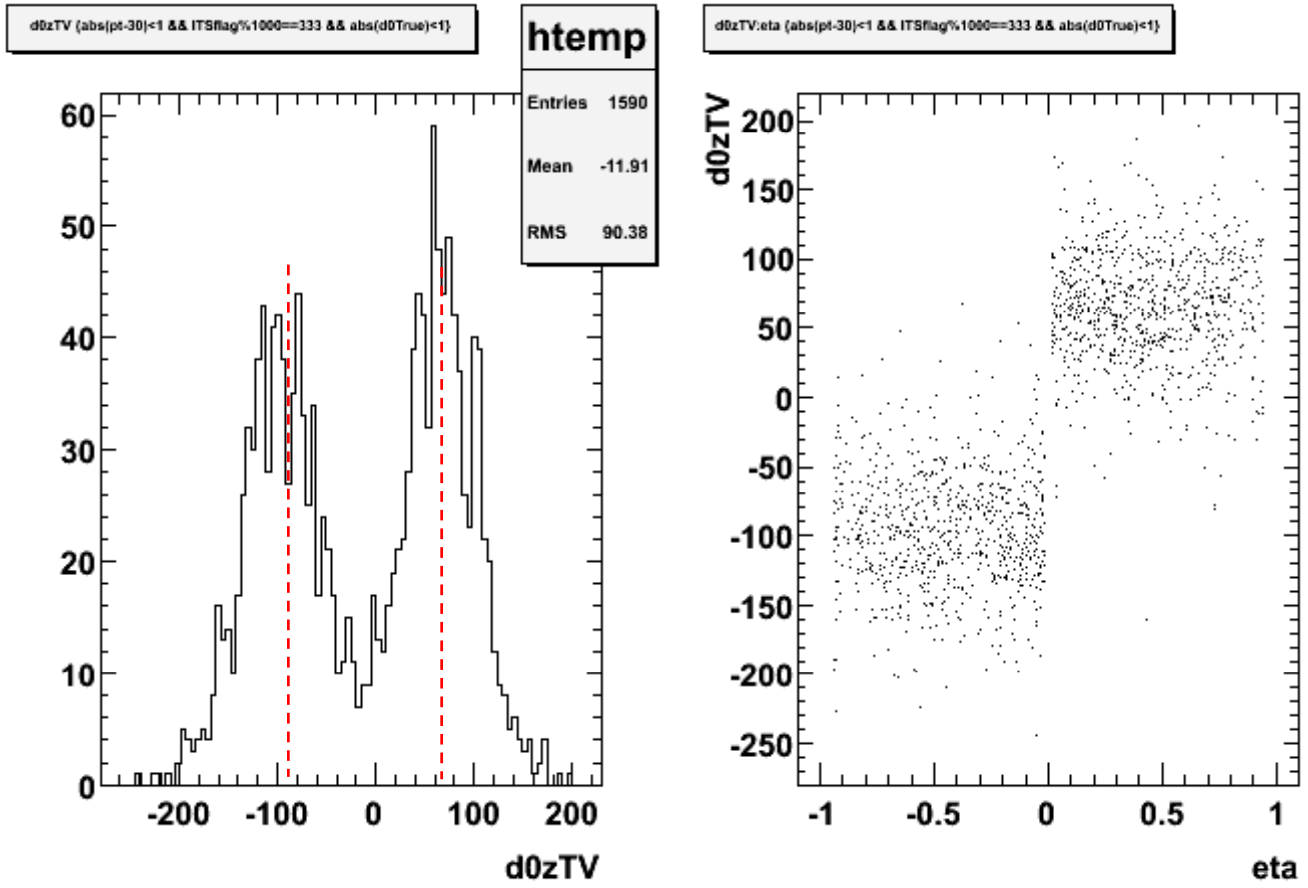
- Since SDD are the most precise layers for z, distribution of track impact parameter along z was affected:



AliRoot HEAD August 2nd, before fix of SDD errors

Issue #1: SDD cluster errors

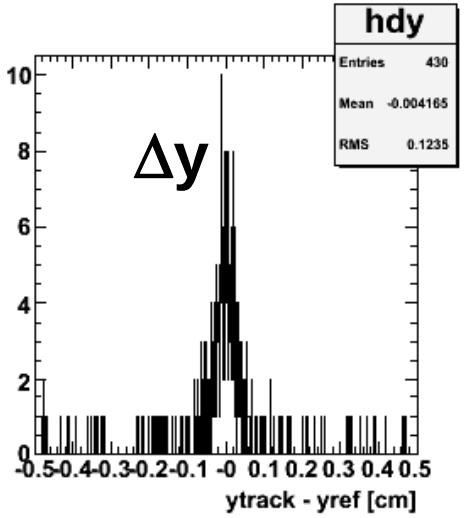
- Since SDD are the most precise layers for z, distribution of track impact parameter along z was affected:

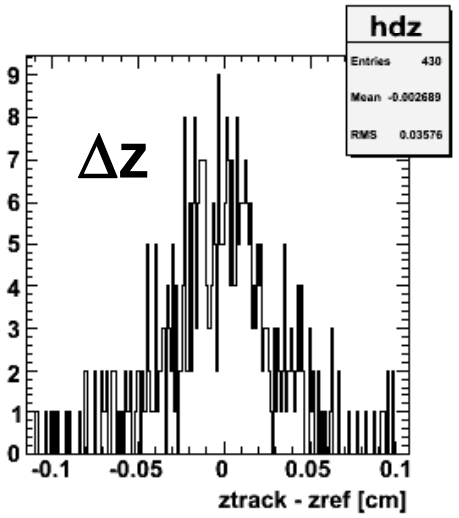


after fix of SDD errors

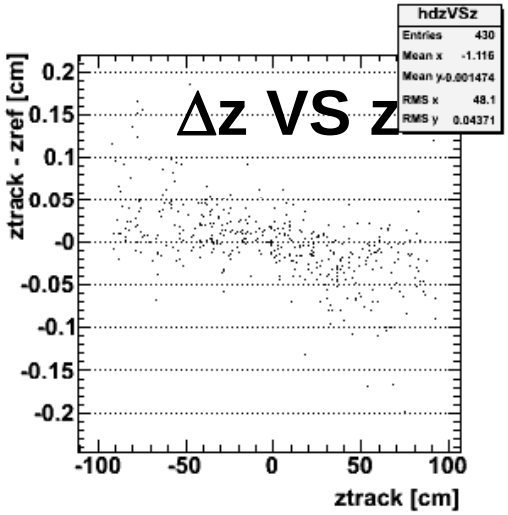
Issue #2: TPC z offset (not time0...)

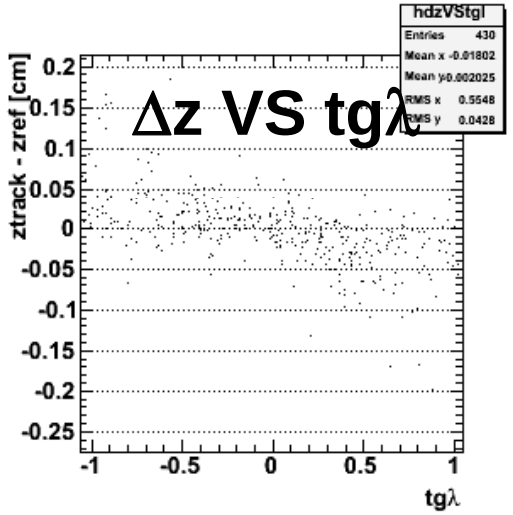
- After some debugging: track VS trackref at TPC inner radius





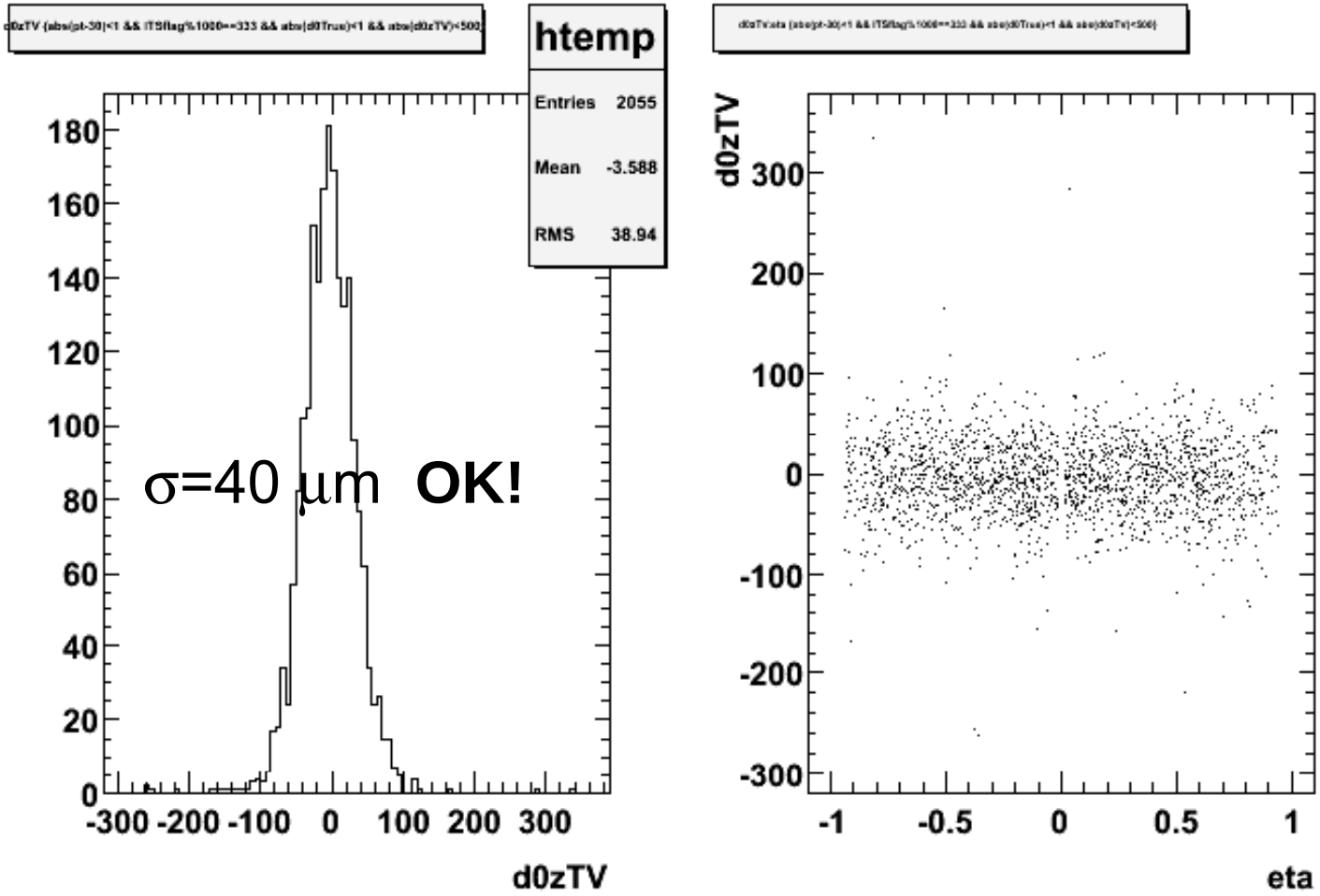
offset of about 0.8 mm, in the same direction as time0 offset, but much larger





now corrected by Marek and Marian: bug introduced when including realistic TPC geometry

Impact parameter res. in z



Current AliRoot HEAD, after all fixes

Material correction: Foreword.

AliExternalTrackParam::CorrectForMeanMaterial (with Andrei, Marian, Yura)

- As in other trackers, material correction done with **AliExternalTrackParam::CorrectForMaterial(d,x0,mass)**, where d is x/X0 and X0 is the rad. length · ρ (in g/cm^2)
- When track crosses material, we have to correct for:
  - multiple Coulomb scattering: using equivalent x/X0
  - ionization energy loss: using equivalent x·ρ
- What does equivalent mean for a composite layer?
  - equivalent x/X0: Σ_i $(x/X0)_i$
  - equivalent x·ρ: Σ_i $(x\cdot\rho)_i$
- While OK for a single-material layer, the variables of CorrectForMaterial are **not OK** for composite layers
- Introduced new **AliExternalTrackParam::CorrectForMeanMaterial(xOverX0,xTimesRho,mass)**, which is always correct (single-material & composite)
- Andrei wrote new (and faster) AliTracker::MeanMaterialBudget
  - TGeo material between to space points
  - returns correctly defined equivalent x/X0 and x·ρ

Material Correction in AliITStrackerMI

- Correction done during the 3 “phases” of tracking

Custers2Tracks →kITSin	PropagateBack →kITSout	RefitInward →kITSrefit

  - We don’t need always the same precision (TGeo is slow!)
    - implement different material budget “modes”
- 10 “layers” of material which we correct for:

TPC->ITS material	1 place in the code
6 ITS layers	3 places in the code
2 shields (SPD & SDD)	3 places in the code
beam pipe	4 places in the code

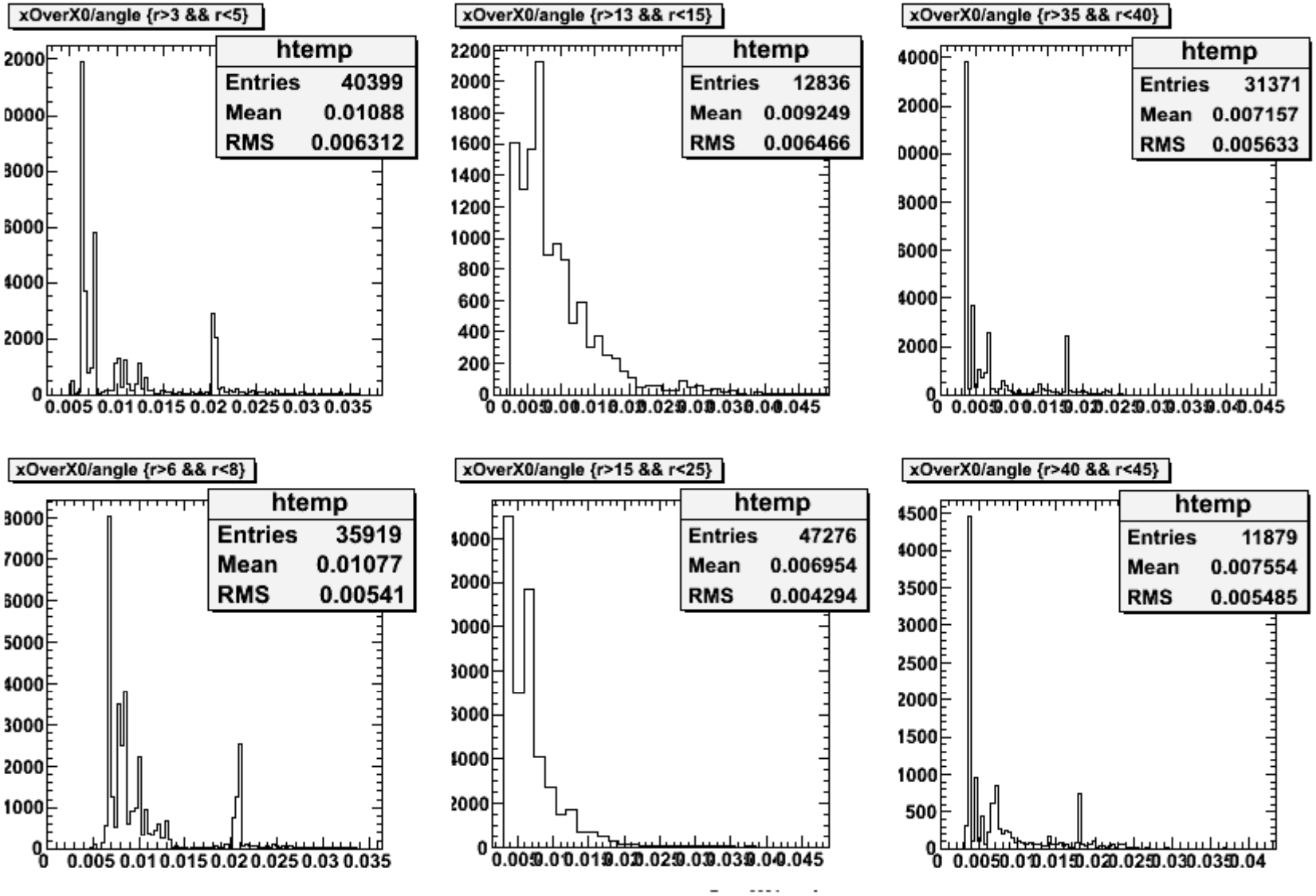
  - Material corrections all around, code multiplication, each time several if’s etc.. to choose if using TGeo or not..

Material Correction in AliITStrackerMI

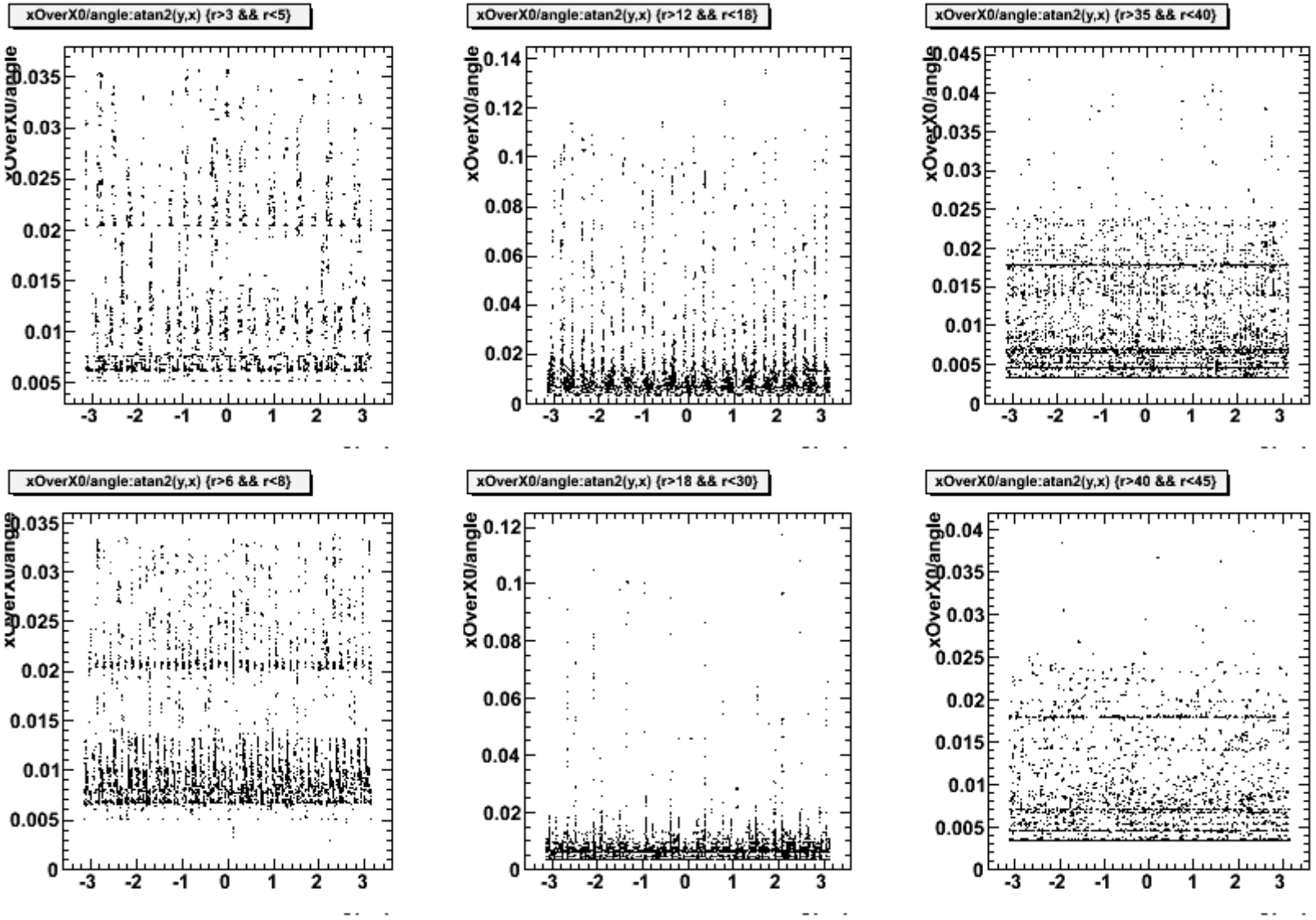
- Material correction grouped in 4 methods:
  - CorrectForTPCtoITSMaterial (material from TGeo)
  - **CorrectForPipeMaterial (called 4 times)**
  - **CorrectForShieldMaterial (called 3 times)**
  - **CorrectForLayerMaterial (called 3 times)**
- For the last 3 methods (pipe, shields, layers), 4 different ways to get the budget implemented:
  a) use “Marian’s” hard-wired numbers (still default)
  b) use TGeo
  c) use a layer-level look-up table with x/X0 and $x \cdot \rho$ for each of the 9 “layers”
     - table is built on-fly using TGeo (no hard-wired numbers!), or could be stored in OCDB
  d) use TGeo to get the material for first prolongation of a track to a “layer”, then use the same material for all prolongations in the hypothesis tree (track-level look-up table)

(not on CVS yet)

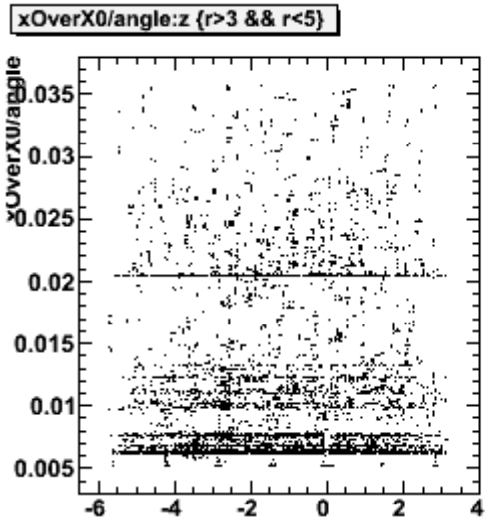
Material plots: x/X0

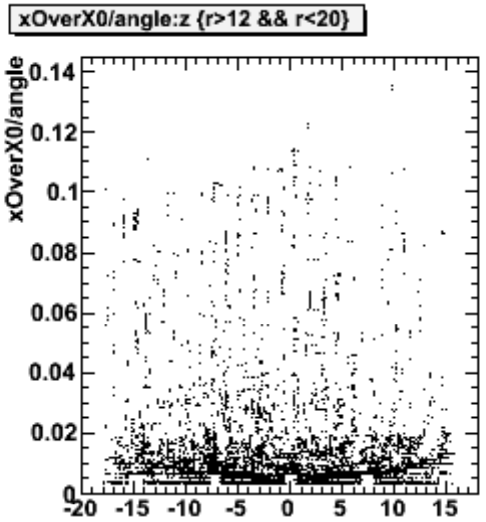


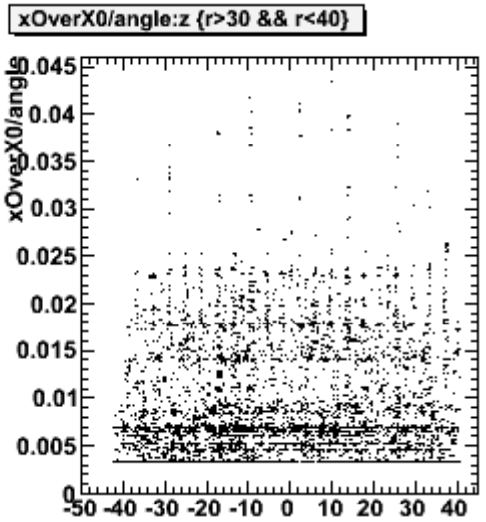
Material plots: x/X0 vs ϕ

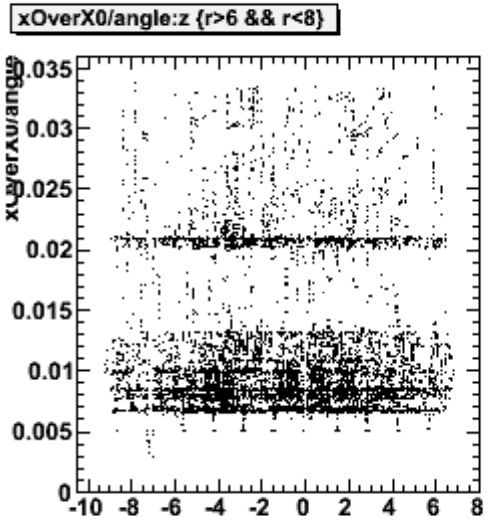


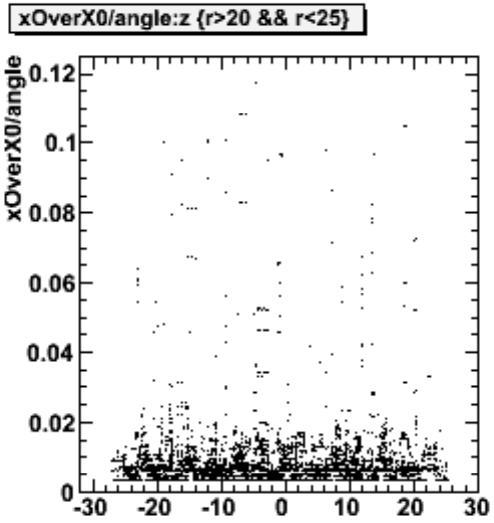
Material plots: x/X0 vs z

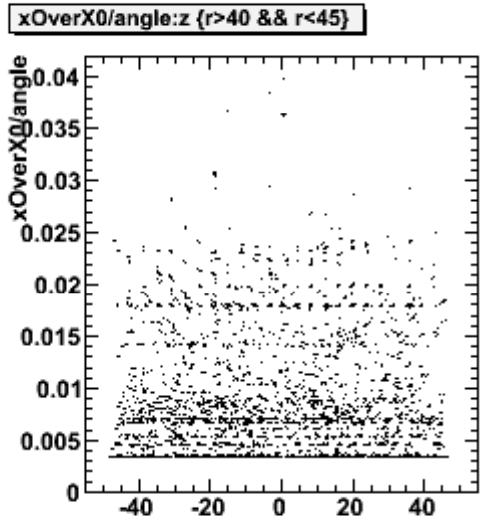






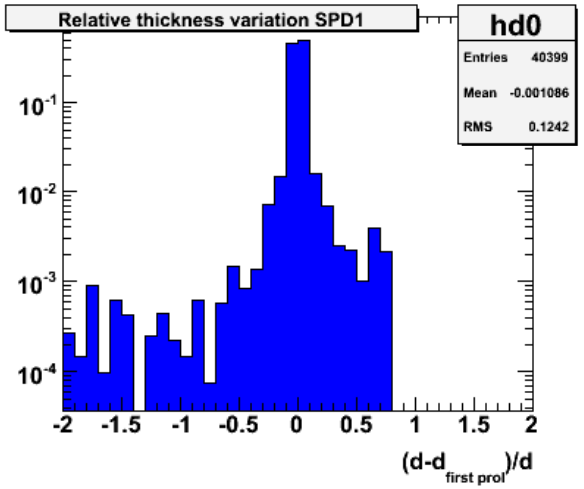


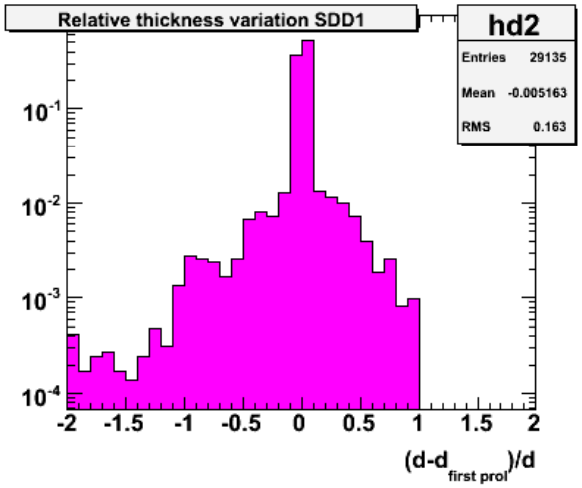


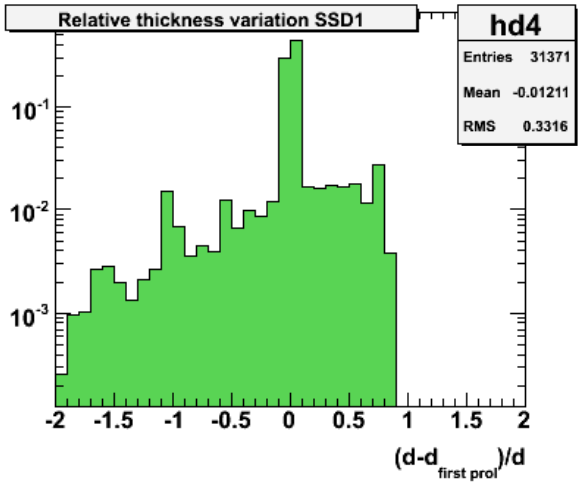


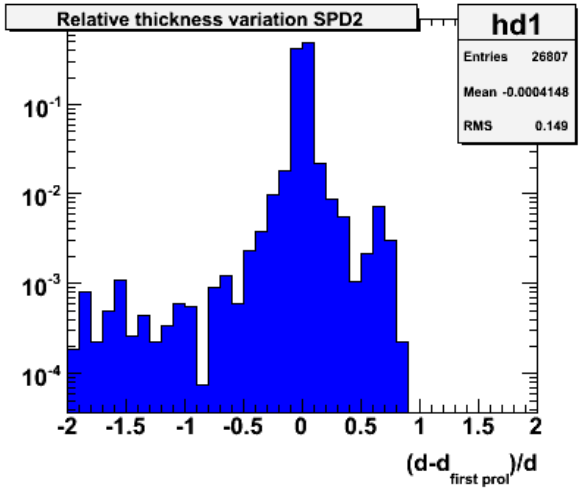
Material plots: x/X0 variation per track

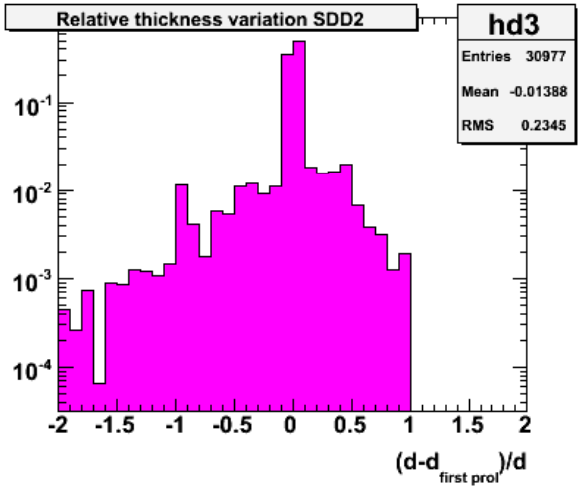
1 test event with $dN_{ch}/dy=4000$

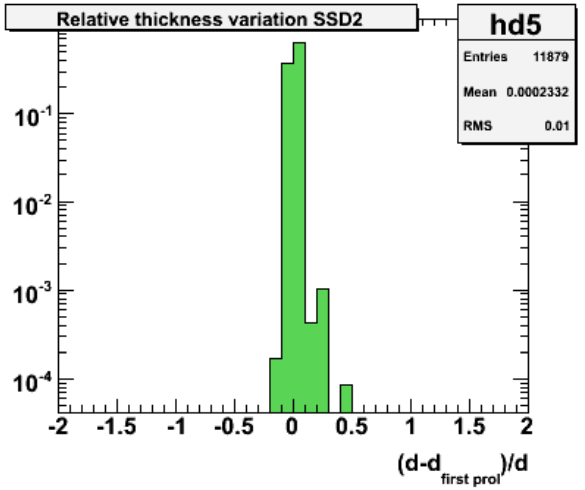












→ **<<1% of prolongations cross material different by more than few 10% with respect to first prolongation**

Material Correction in AliITStrackerMI

- During each tracking phase we can use a different method to get the material
- Everything steered by AliITSRecoParam::SetUseTGeoInTracker(Int_t tgeo)
- Currently, 5 possible settings:
  - tgeo=0 → hard-wired numbers
  - tgeo=1 → always TGeo
  - tgeo=2 → always layer-level TGeo look-up table
  - tgeo=3 → track-level TGeo look-up table during INWARD and OUTWARD, normal TGeo during REFIT
  - tgeo=4 → track-level TGeo look-up table during INWARD and OUTWARD, layer-level TGeo look-up table during REFIT

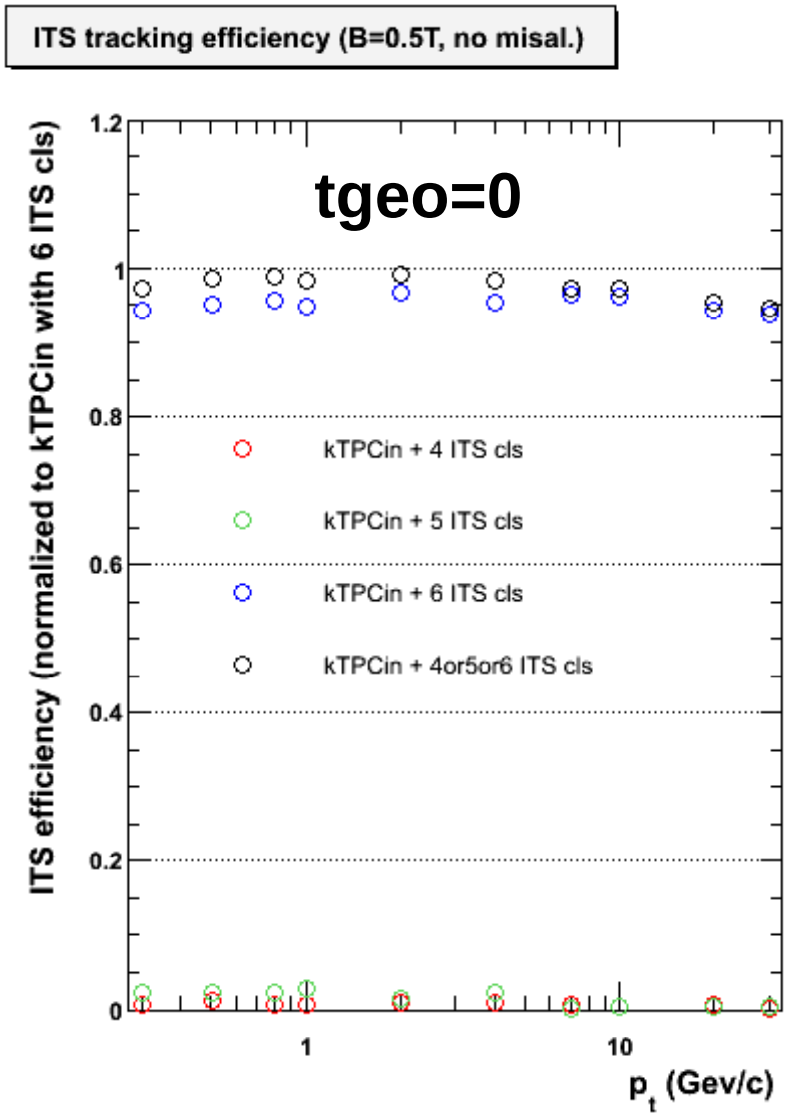
(not on CVS yet)

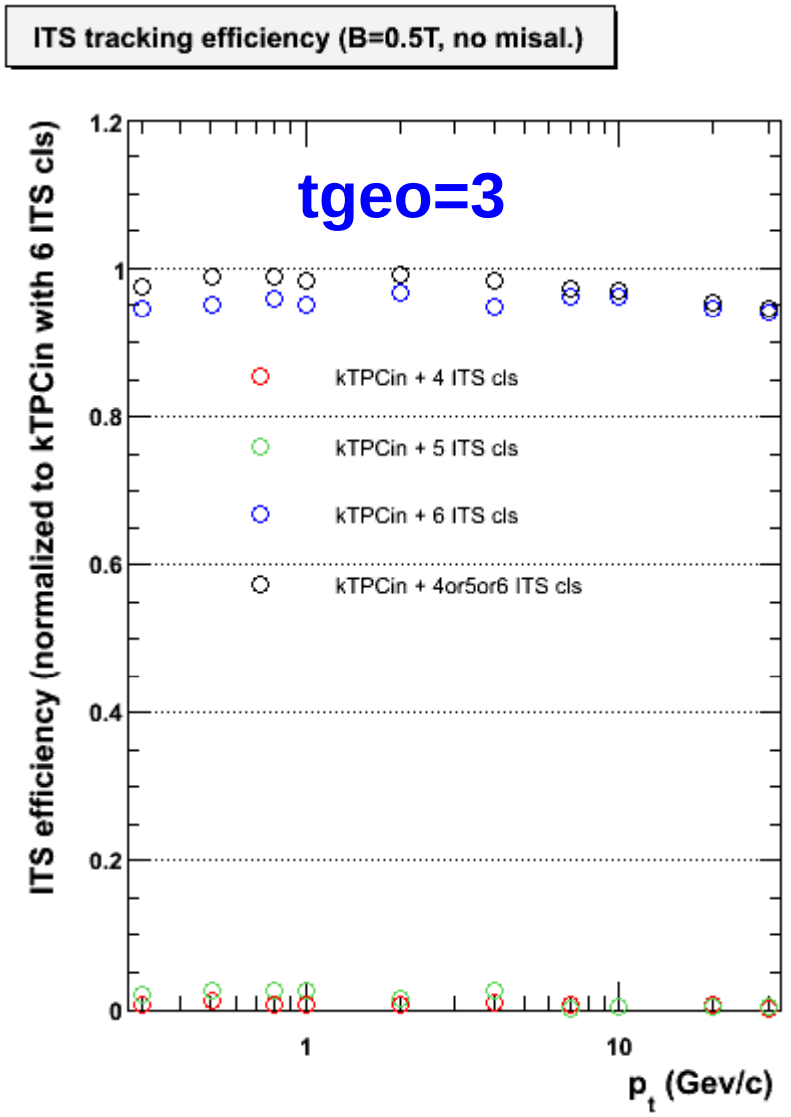
Getting the material from TGeo
CPU time test

dN_{ch}/dy =	10	100	1000	4000
kITSin CPU (s) [only AliITStrackerMI, no AliITStrackerSA]	**0.06** **0.92** **0.06** **0.068** **--**	**0.6** **10.5*** **2.2*** **2.5*** **2.5***	**7.0** **111.5*** **8.6*** **12.4*** **11.7***	**47** **707*** **49*** **63*** **--**
Tracking CPU (s) [AliTPCtrackerMI + AliITStrackerMI]	**0.92** **1.88** **0.96** **0.97** **--**	**5.5** **16.0*** **7.1*** **8.2*** **7.4***	**53.6** **163.7*** **54.1*** **63.9*** **57.5***	**228** **913*** **244*** **271*** **--**
kTPCin CPU (s)	0.07	0.8	9.9	52

* -1.6 s for layer-level look-up table (once per session)

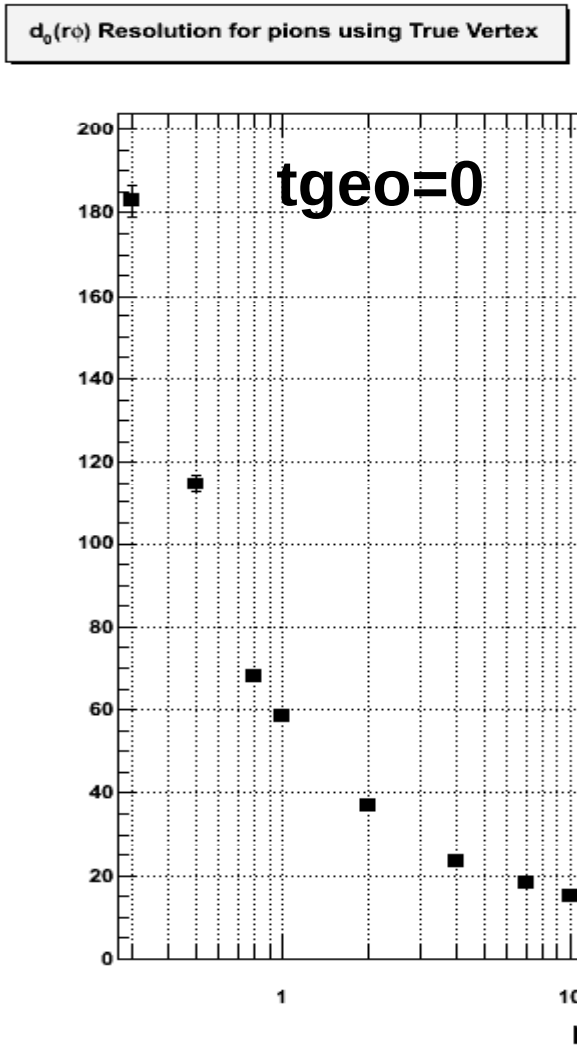
Getting the material from TGeo
ITS tracking efficiency test

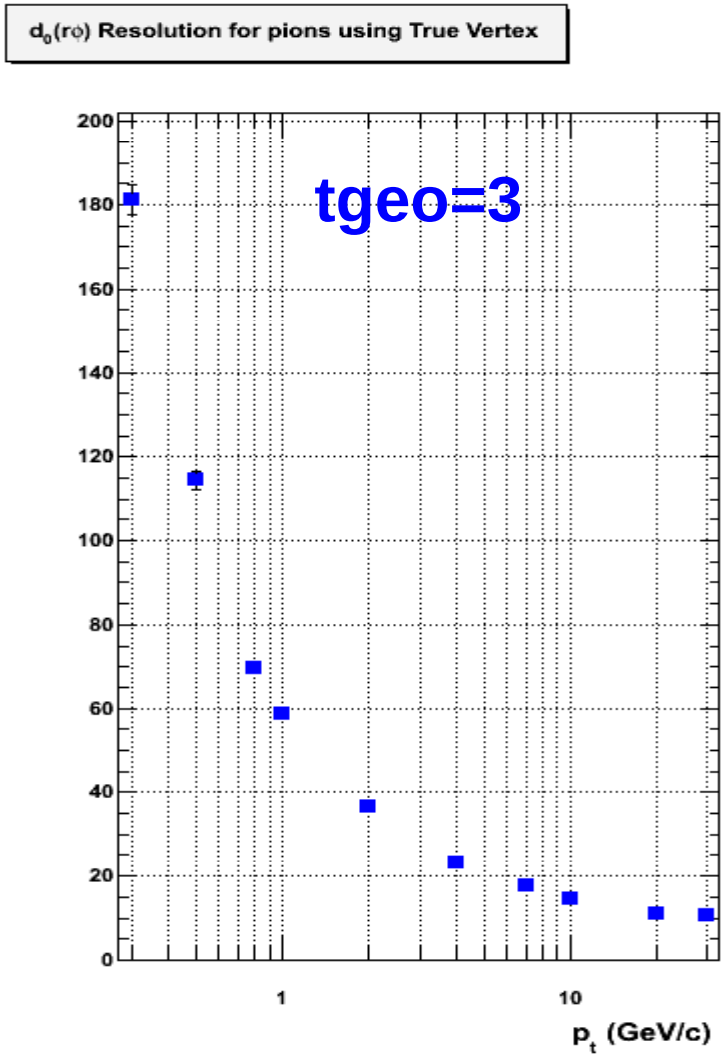




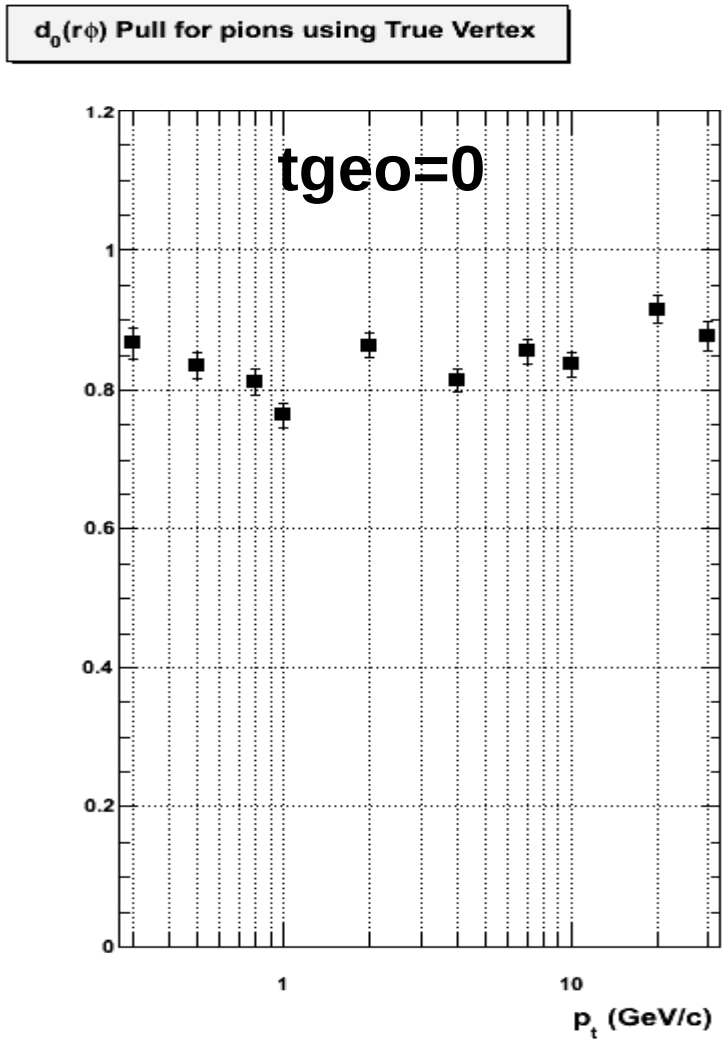
Getting the material from TGeo

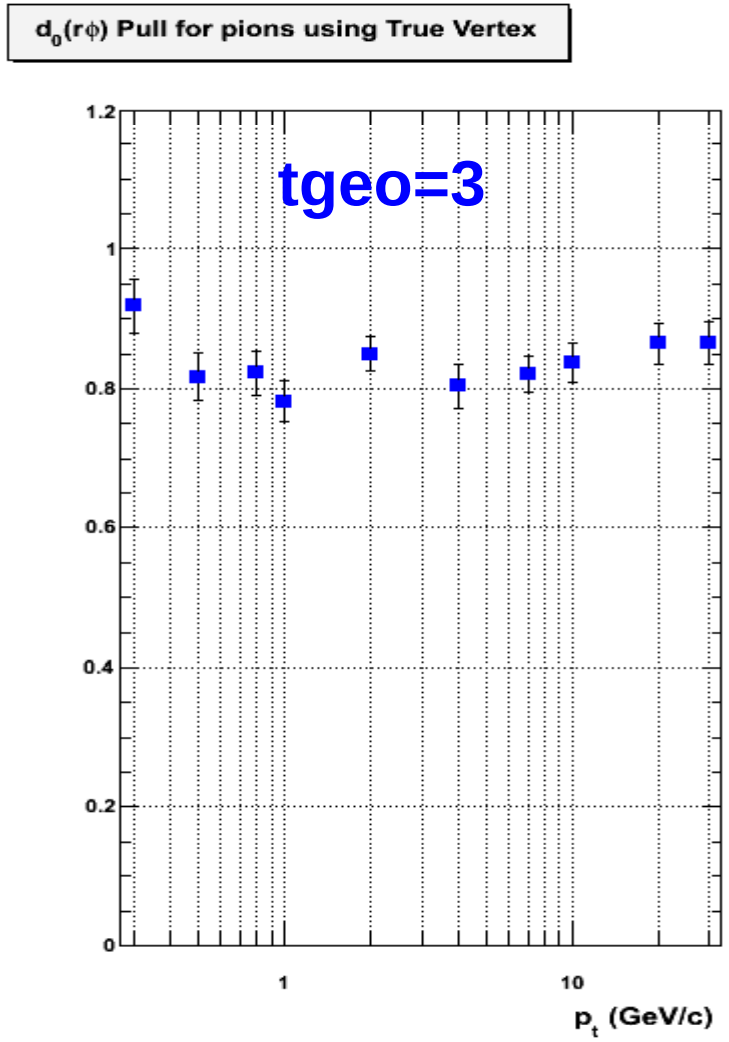
d_0 resolution test



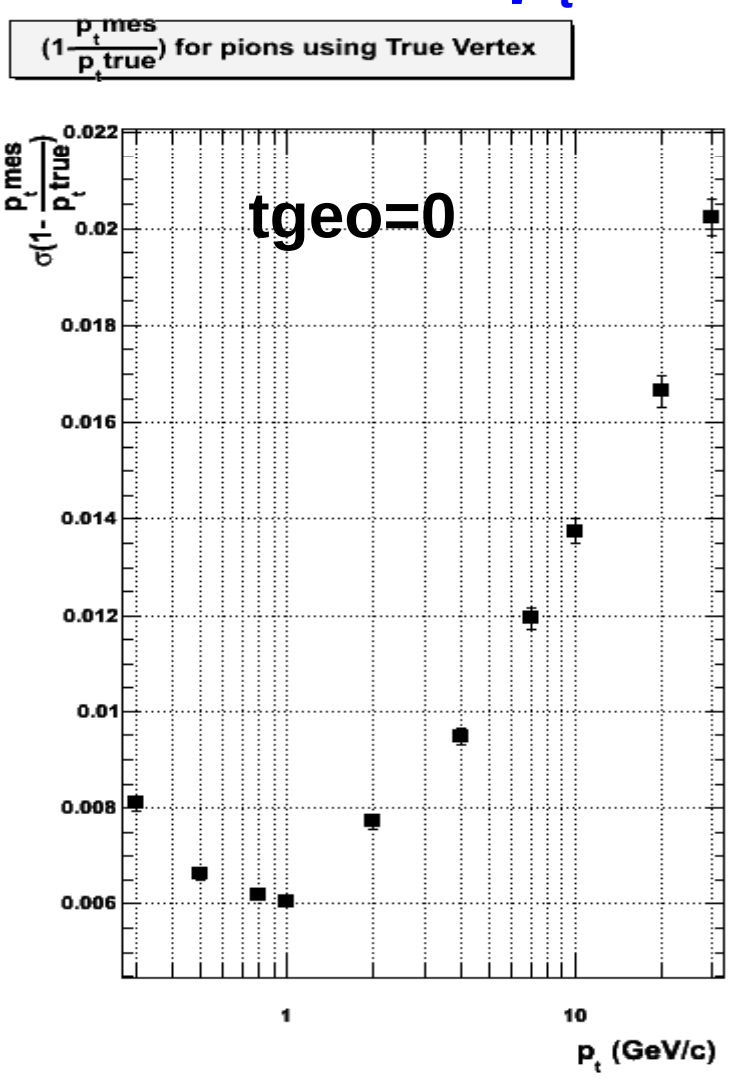


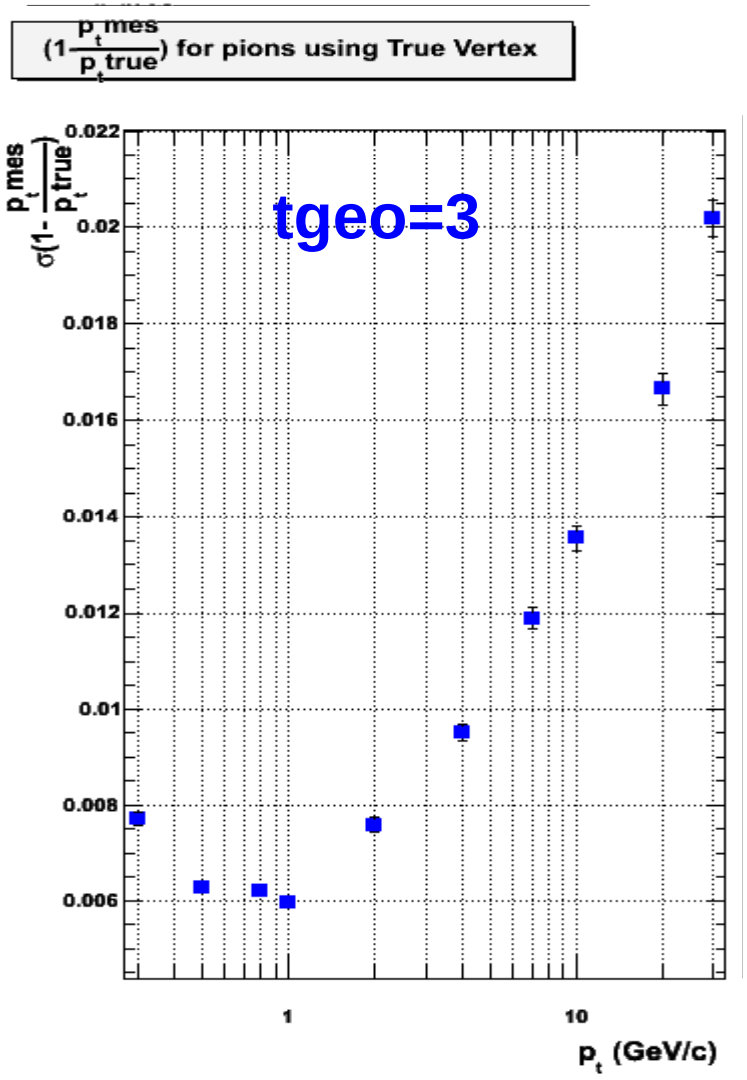
Getting the material from TGeo
d_0 pull test



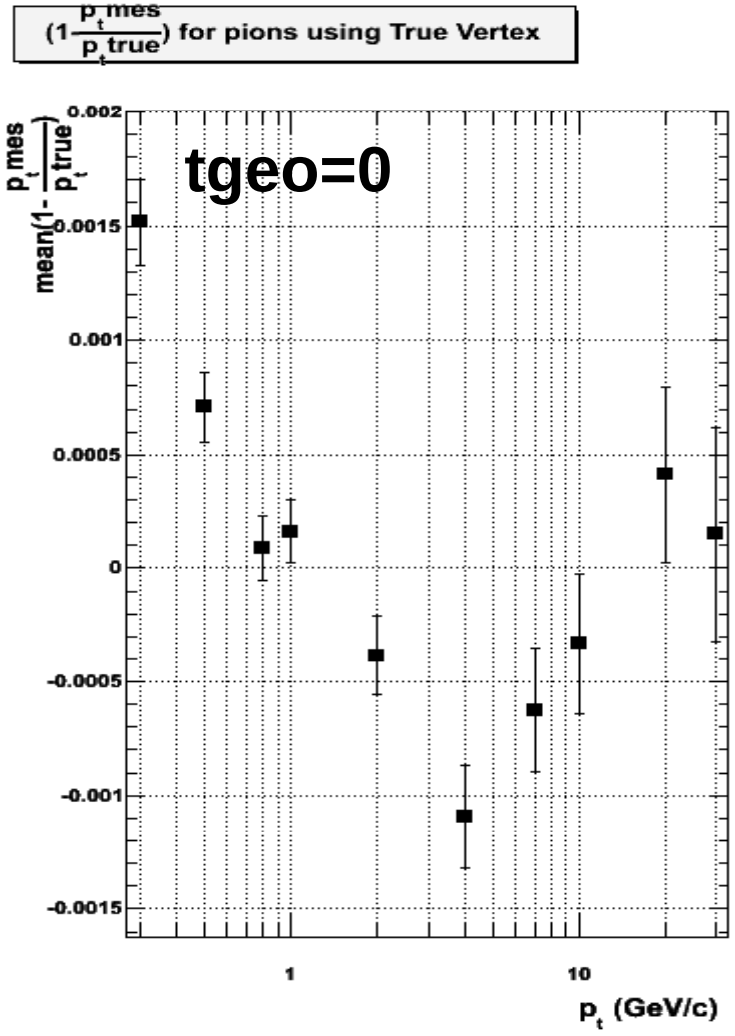


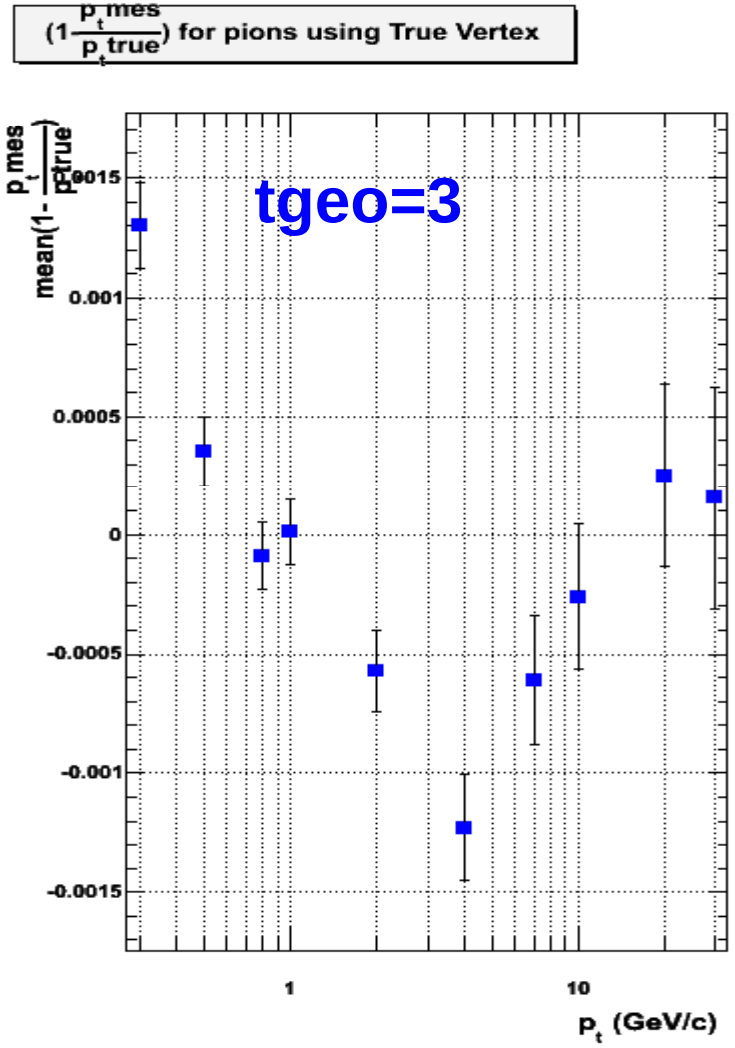
Getting the material from TGeo
p_t resolution test





Getting the material from TGeo
p_t resolution test (mean of $\Delta p_t/p_t$)





to be investigated ...

Summary

- Factorization is progressing (only for trackerMI)
  - faster obsolescence of trackerV2 (not good...)
- AliITSClusterParam activities started
  - long way to go here, need strategy to extract cluster params
- Material from TGeo
  - major clean-up
  - identified strategy for best balance between time, performance and removal of hard-coded numbers
- Few important bugs discovered and corrected
  - last one this morning (ITSClusterMap mismatch spotted by F.Prino)
- Open points
  - tracker has to be made decalibration and misalignment aware
    - these info has to go to the tracks' covariance matrix
  - AliITStrackerSA is very slow (see Marian's blacklists)