Reconstruction
and code development

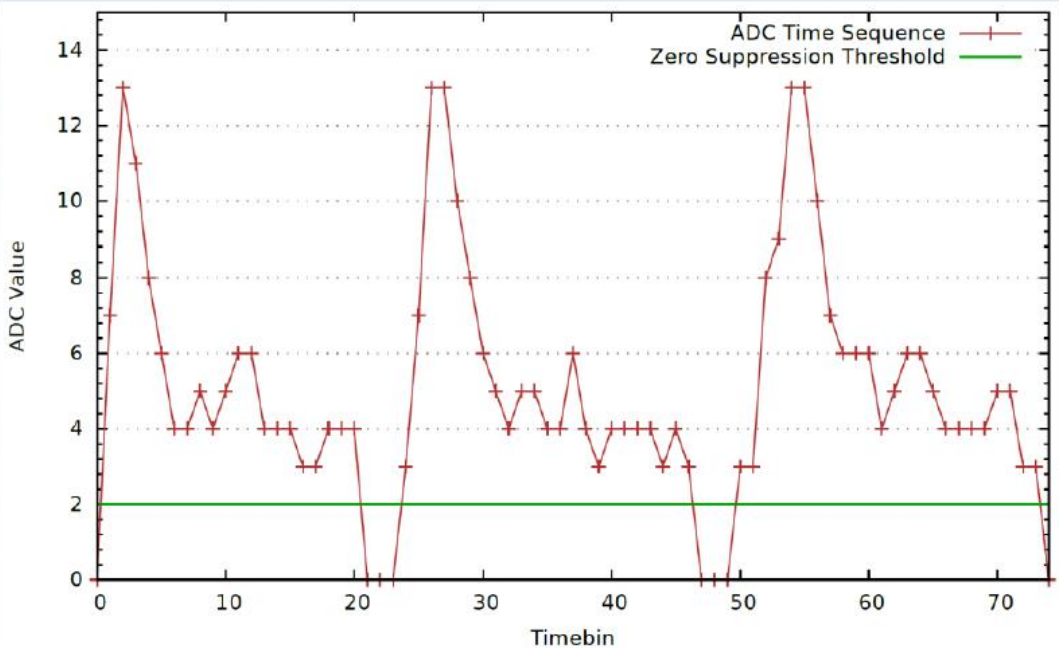
HLT cluster finder (emulator) validation

Main goal: reduction of split clusters due to the noise

Current Problems with Split Clusters

FIAS Frankfurt Institute for Advanced Studies

- **Problem: after the peak, there is a ramp down until the ADC value drops below zero-suppression. (See below, consider readout is in inverse time direction.)**
  - Noise in the TPC overlaps the ramp, leading to **small fake-peaks, and possibly to fake clusters.**
  - This was not seen in run 1, because of lower gain, where the ramp was below zero-suppression threshold.
- **HLT ClusterFinder is 1D+1D:**
  - First in time, then in pad direction.
  - Each cluster consists of many time sequences.
  - Cluster tagged, if at least one sequence is split.
    - **This is too loose and often incorrect**.
- **Goal: Improve HLT TPC HWCF in order to:**
  - **reject fake clusters**
  - **avoid incorrect tagging**
- **Currently (with HWCF running now, 2016, pp):**
  - **8.3%** of clusters split in pad direction.
  - **30.1%** of clusters split in time direction.



22.09.2016 5

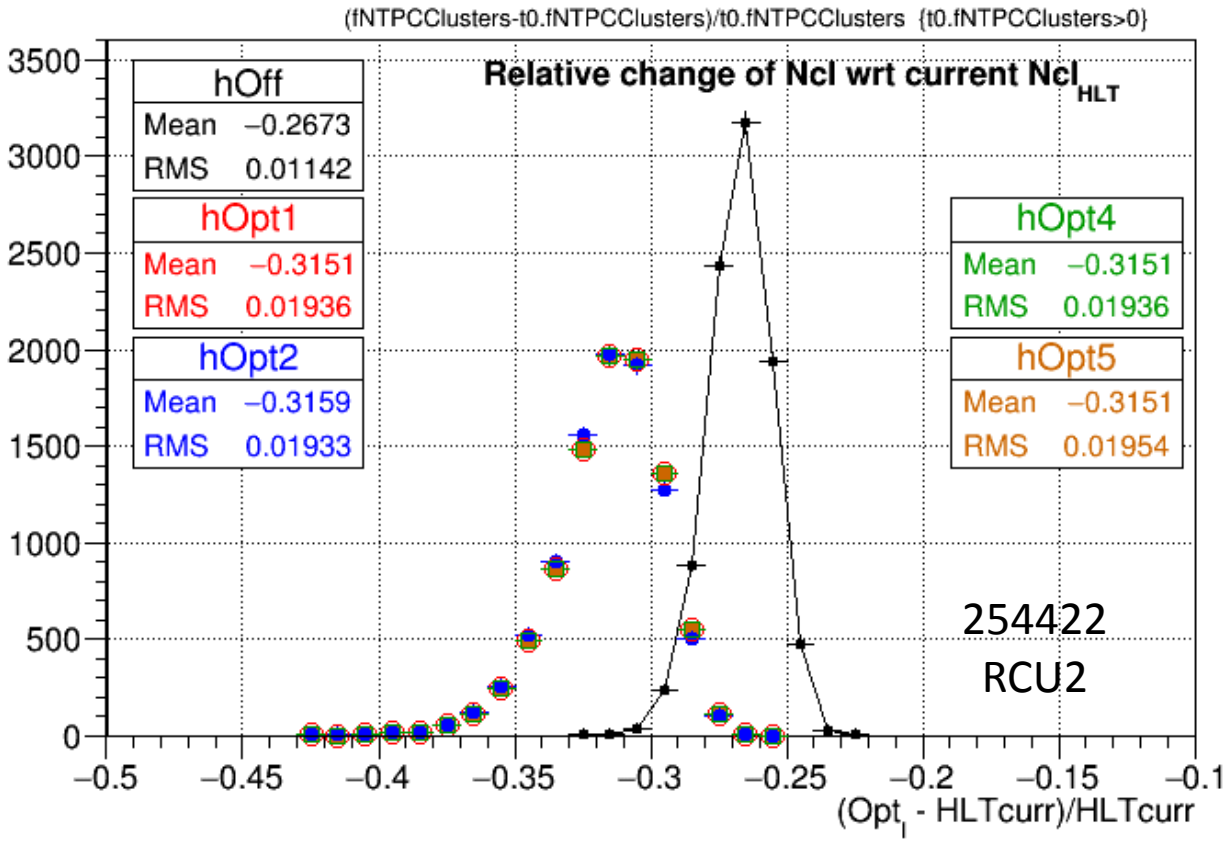
D.Rohr

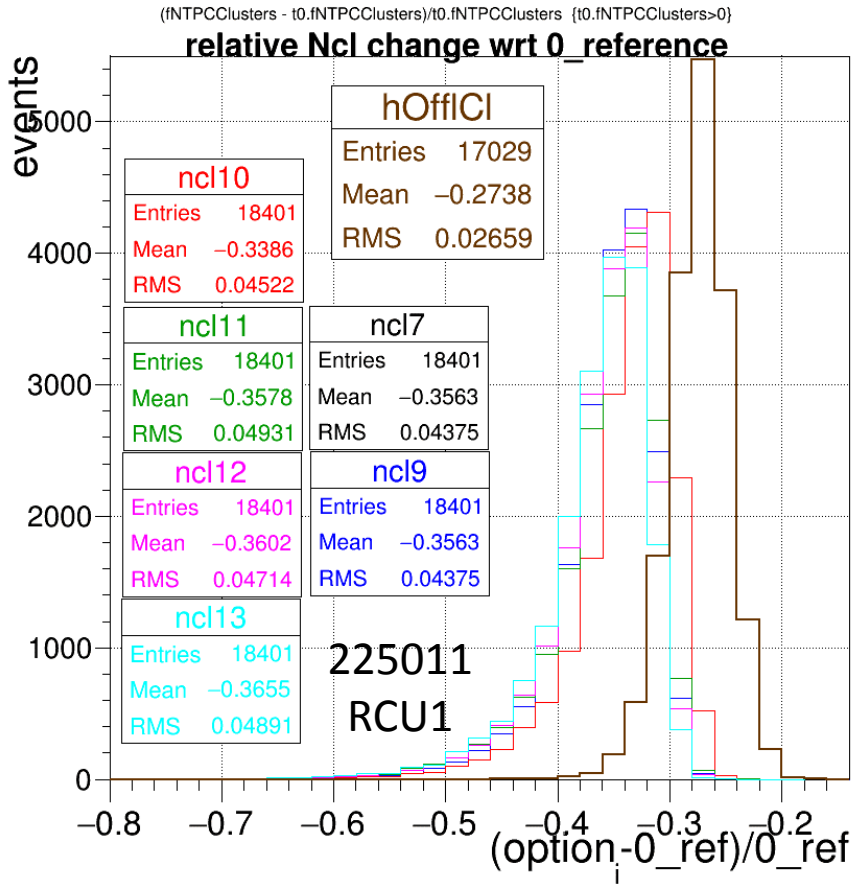
1st step to better TPC data compression (see full presentation for other steps)

Benchmark of performance with modified HLT CF

J.Wiechula, D.Rohr, R.S, C.Zampolli

Same mode A runs from 225011 (LHC15f, 26kHz, RCU1) and 254442 (LHC16h, 130 kHz, RCU2) reconstructed (with cpass0 + SP calibration) with offline CF, standard HLT CF (0_ref) and modified HLT CF with multiple options: agressivity of noise cut, precision of stored charge and cluster shape info, tagging of edge clusters.



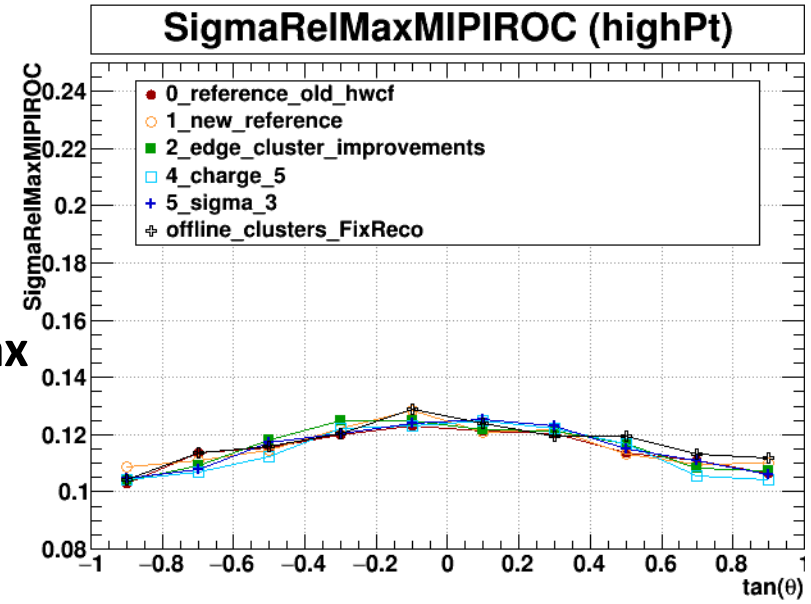


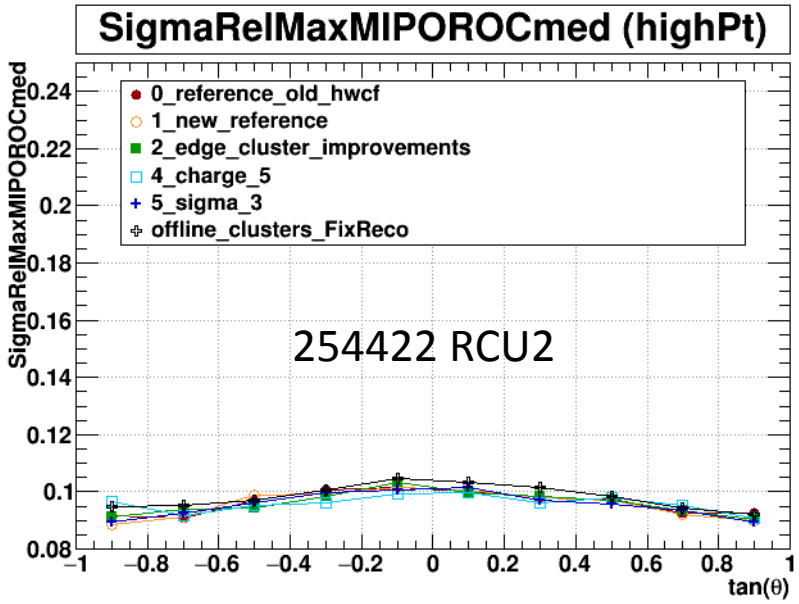
32(highIR)/35% (low IR) reduction of Nclusters with modified CF wrt standard HLT CF (5/8% reduction wrt Ncl offline CF)
Actual reduction in TPC data size will be smaller (~20%) since noise clusters are more compressible.
Total size reduction depends on trigger mix (TPC share)

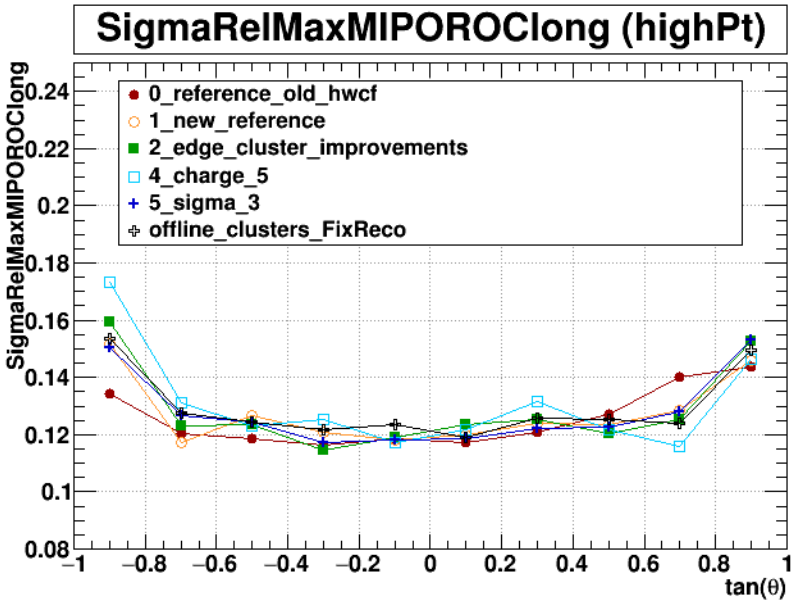
TPC dEdx performance: no difference in resolution

even in CF versions (4,5) with restricted charge info precision

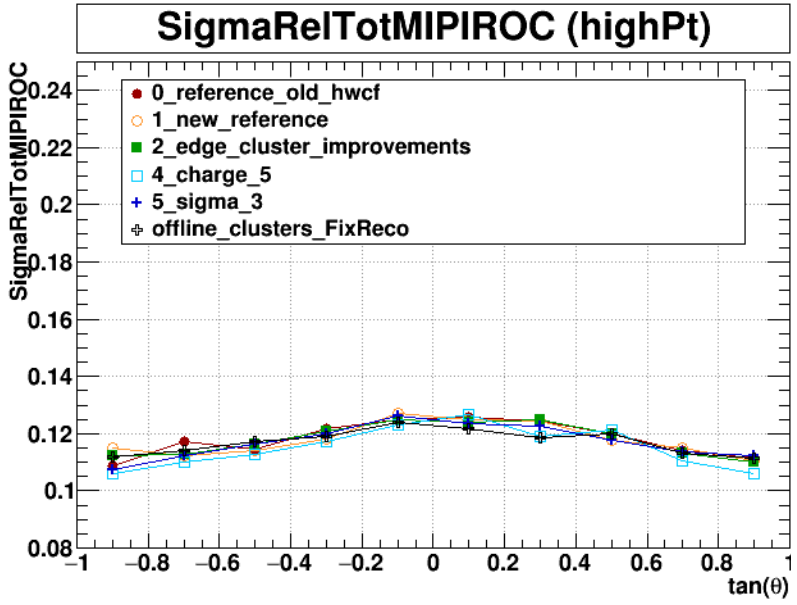
Qmax

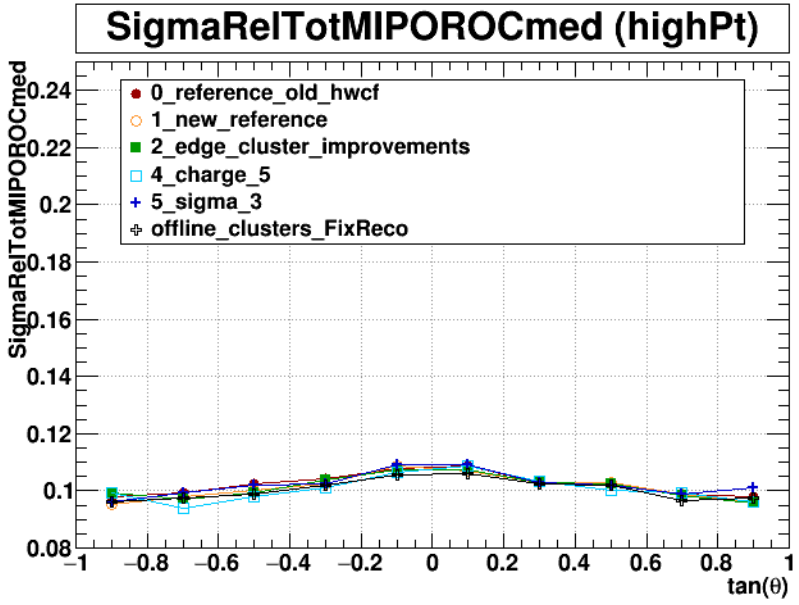


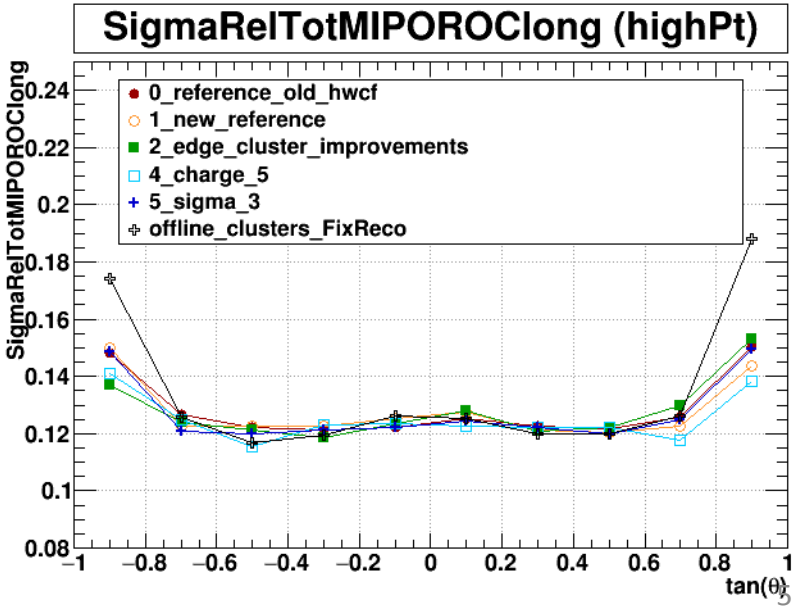




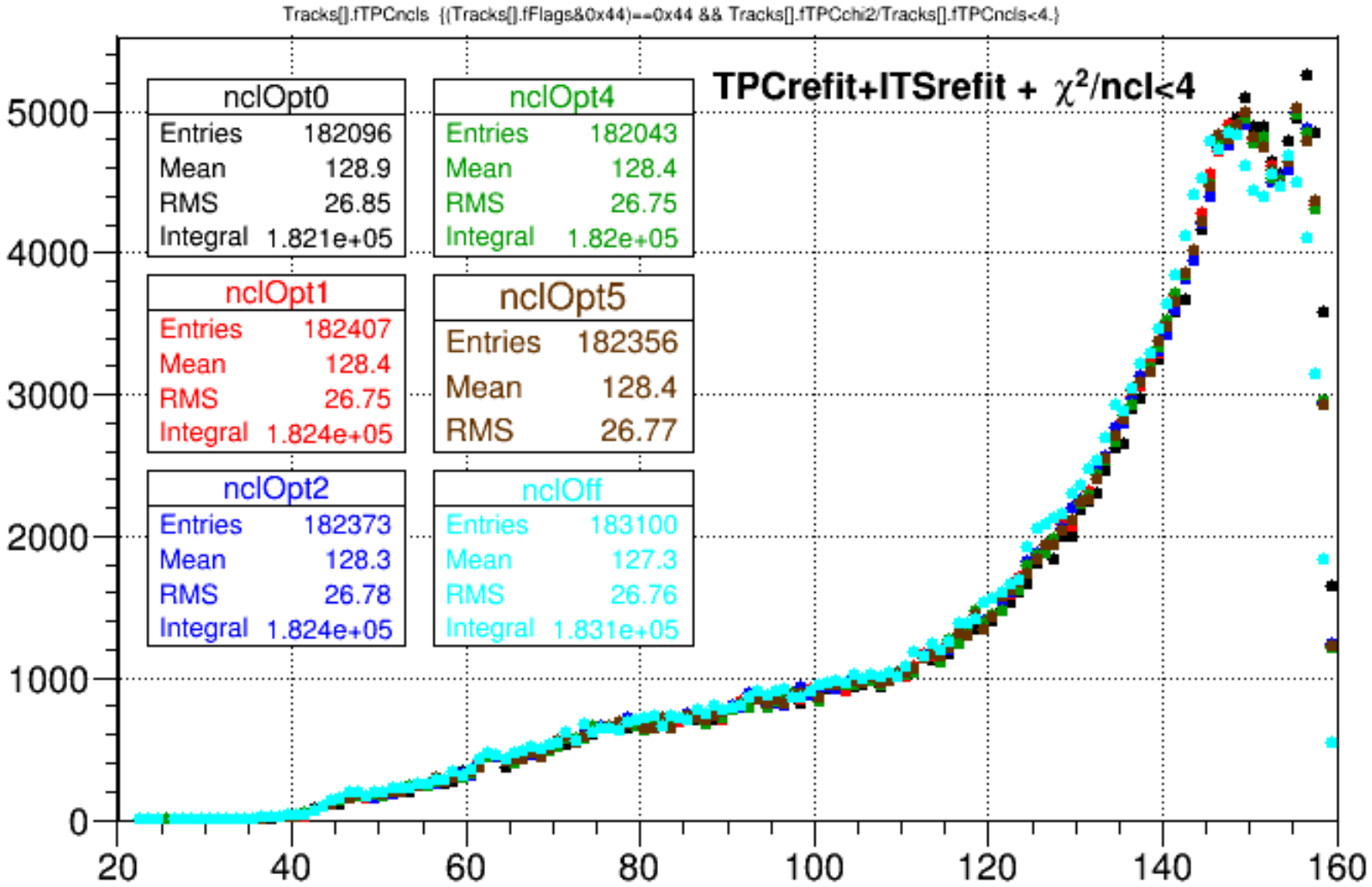
Qtot

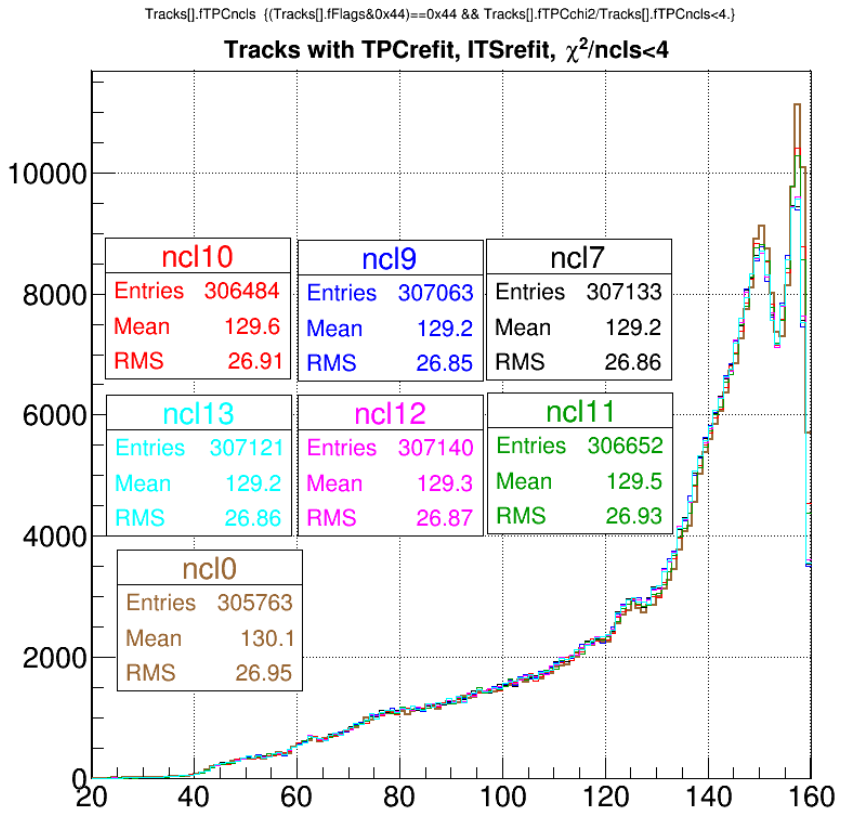






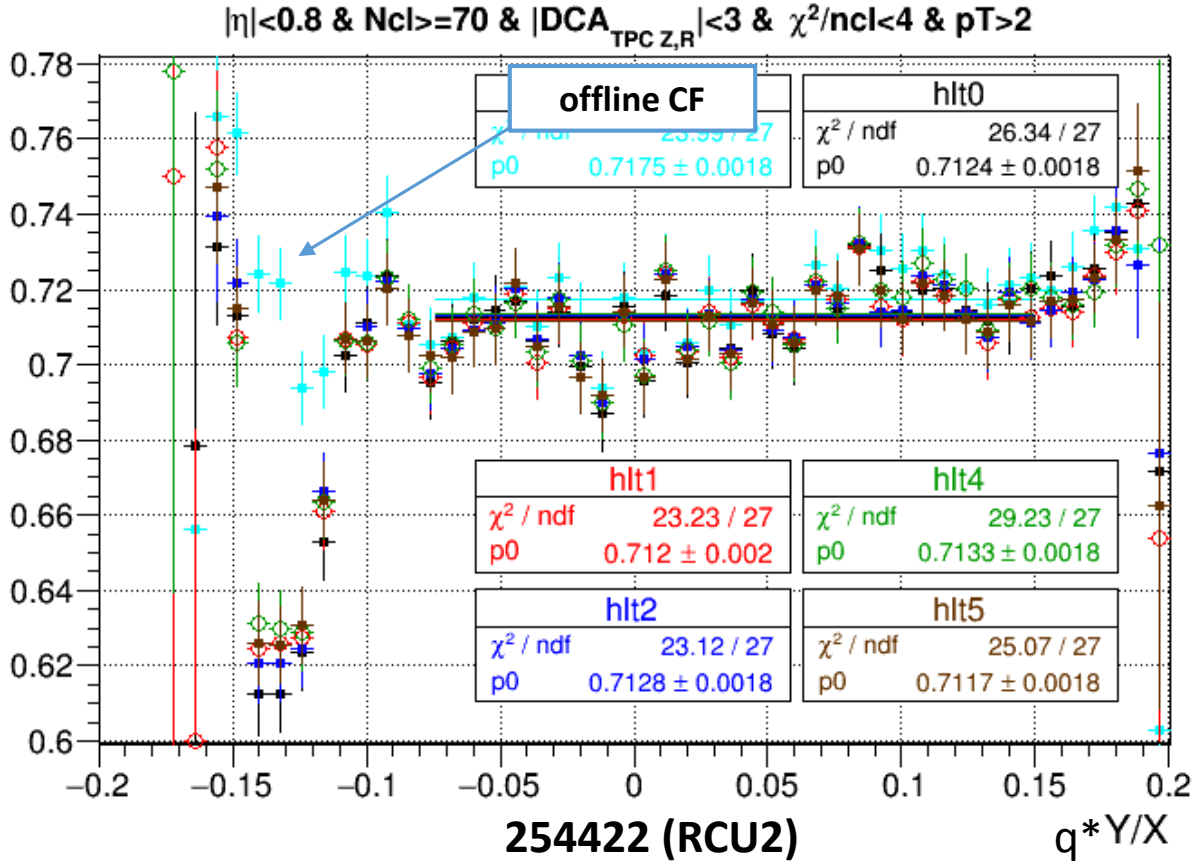
Nclusters/track and N matched tracks (equivalent statistics for all tests)

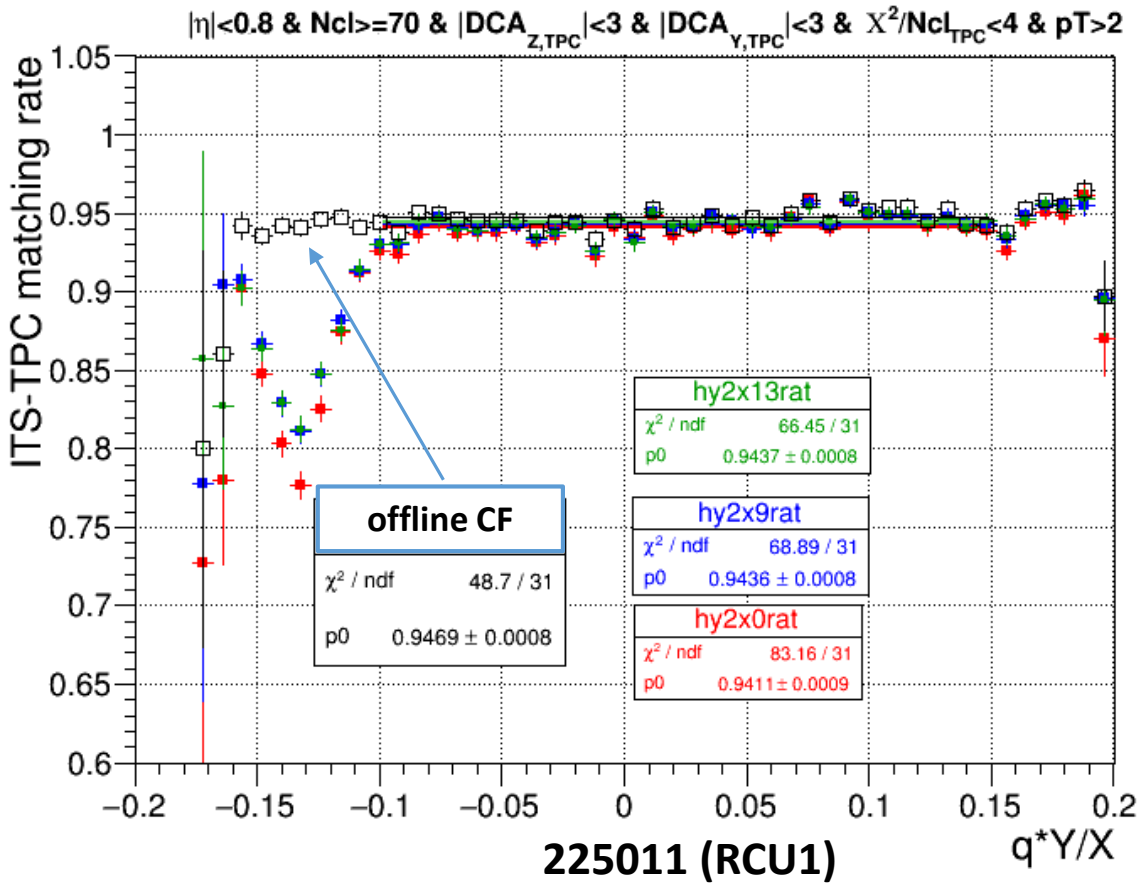




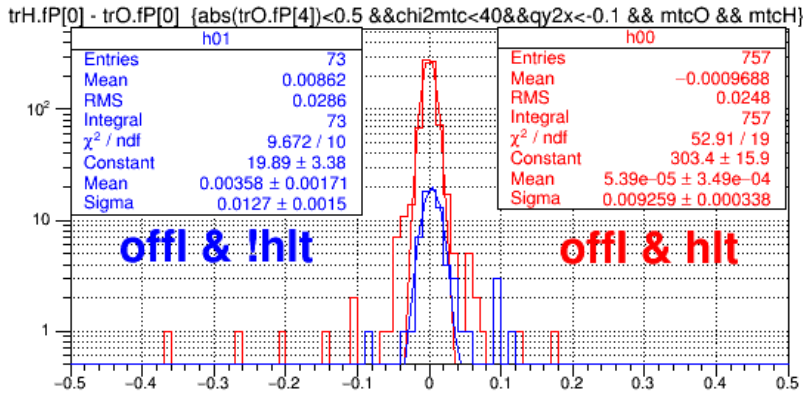
Matching rate (pT>2) vs angle in the sector

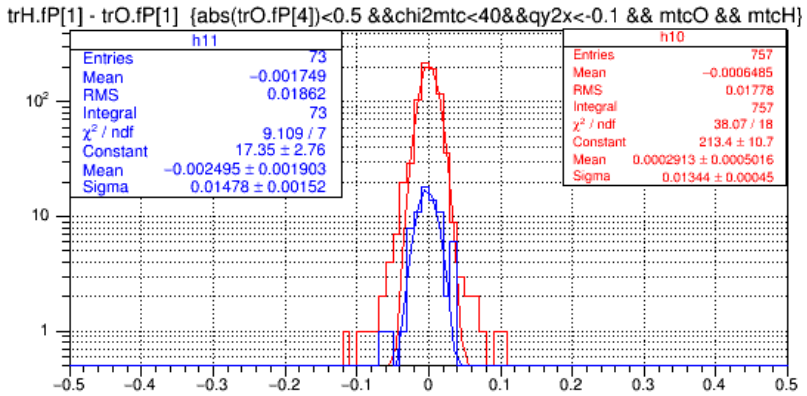
- ~15% depletion of ME on the edge (tracks bending outwards) both in low- and high-ID runs with HLT CF
- Insignificant improvement of matching on the edge with modified HLT CF
- Outside this region matching with all HLT CF versions is consistent with offline CF

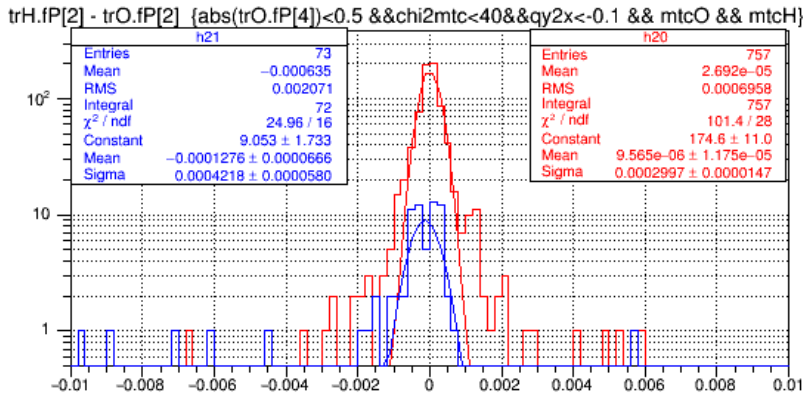


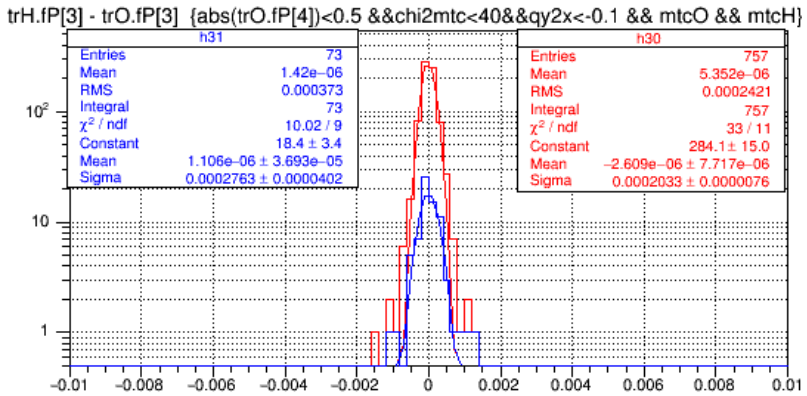


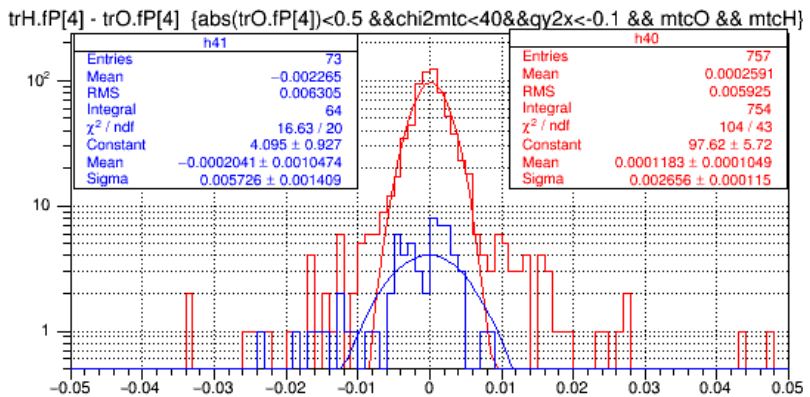
Track-by-track matching between productions with offline and HLT CF











Difference between 5 track params at the TPC inner radius:

Matched both with Offl. and HLT CF

Matched with Offl. but not HLT CF

Param	σΔ		
ΔY	0.13	0.09	$\cdot 10^{-1}$
ΔZ	0.15	0.13	$\cdot 10^{-1}$
Δsinφ	0.42	0.30	$\cdot 10^{-3}$
Δtgλ	0.28	0.20	$\cdot 10^{-3}$
$\Delta q/p_T$	0.57	0.26	$\cdot 10^{-2}$

No bias in parameters difference, no significant outliers

Largest deviation for q/pT (still need to check charge separated)

Cluster angular (q-corrected) position vs pad-row for

tracks matched to ITS both in Offl. and HLT CF prod.

and

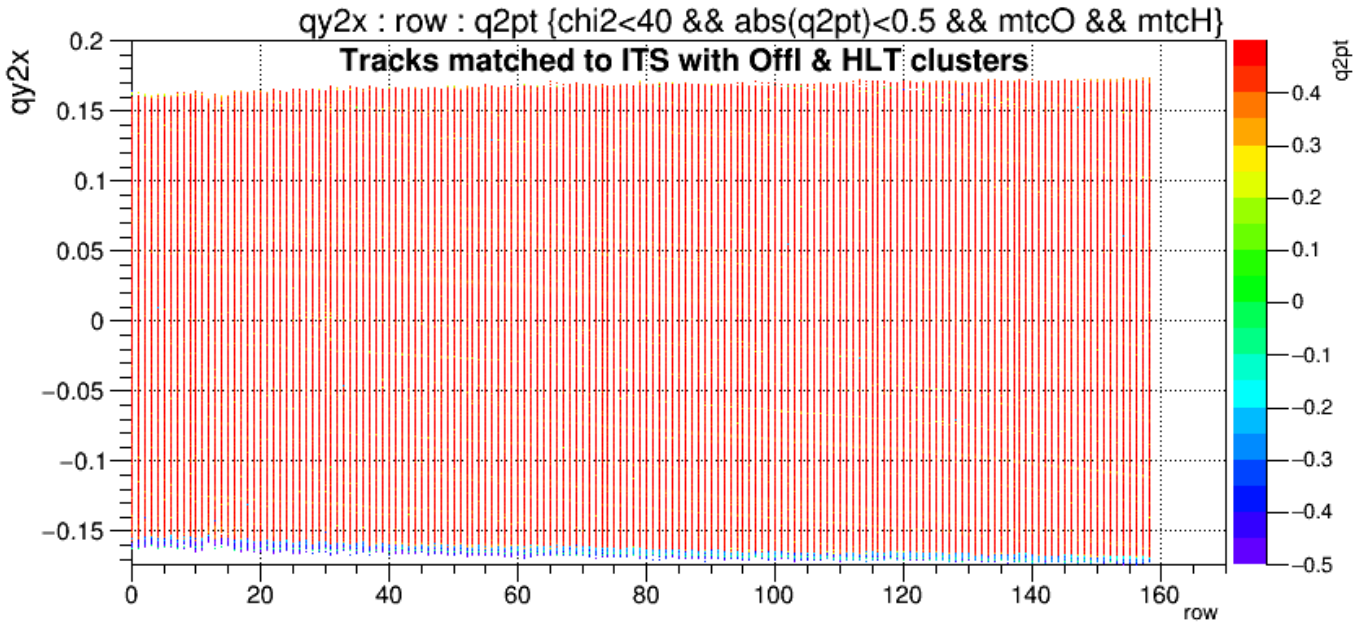
tracks mathced in Offl. but not HLT CF prod.

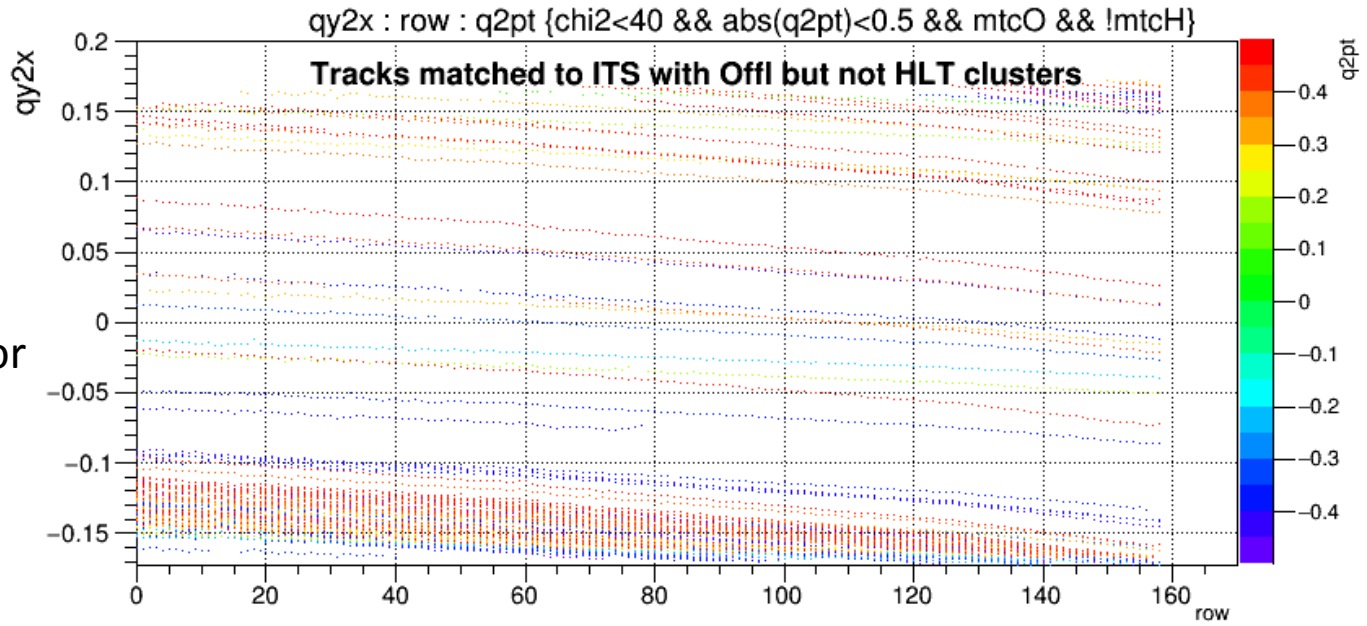
Matching fails due to the deterioration of p_T resolution by truncated edge clusters (at large radii)

Offl. CF compensates edge-cluster positions bias due to the truncation

HLT assigns COG (biased by truncation) as position

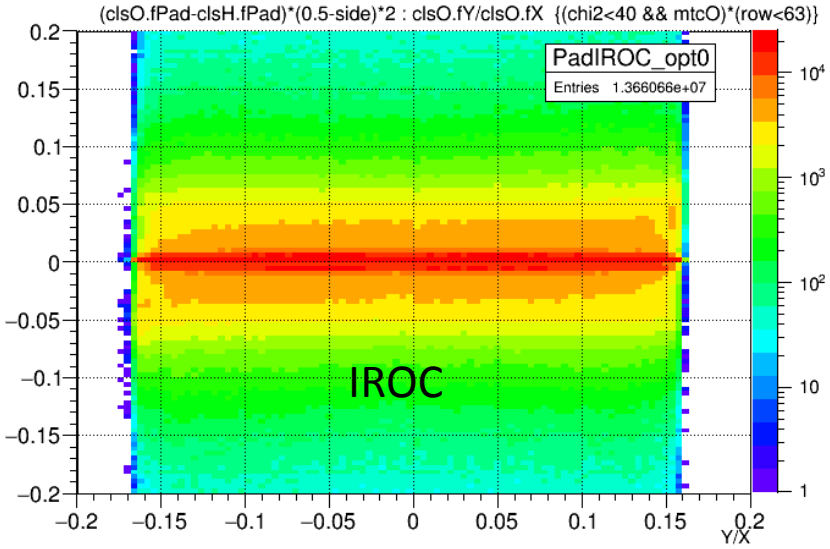
One of the tested HLT CF options (2) tried to account for truncation by assigning position to peaking pad

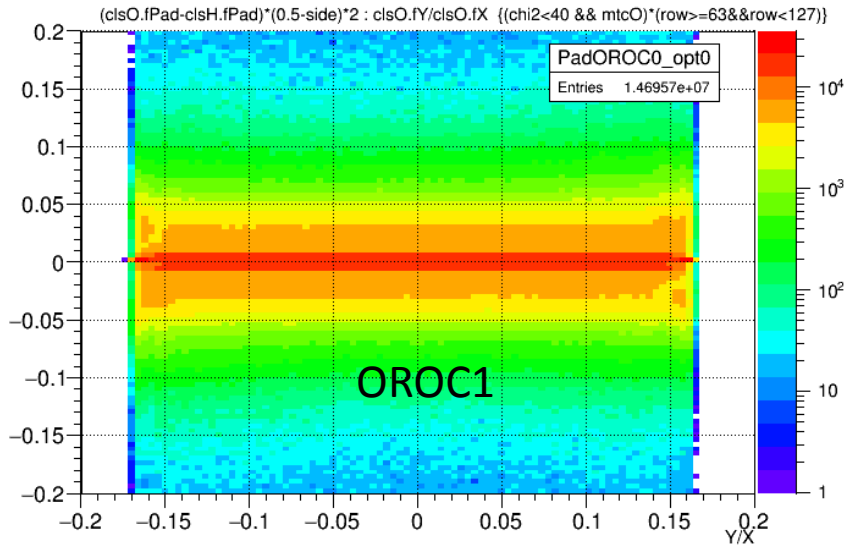


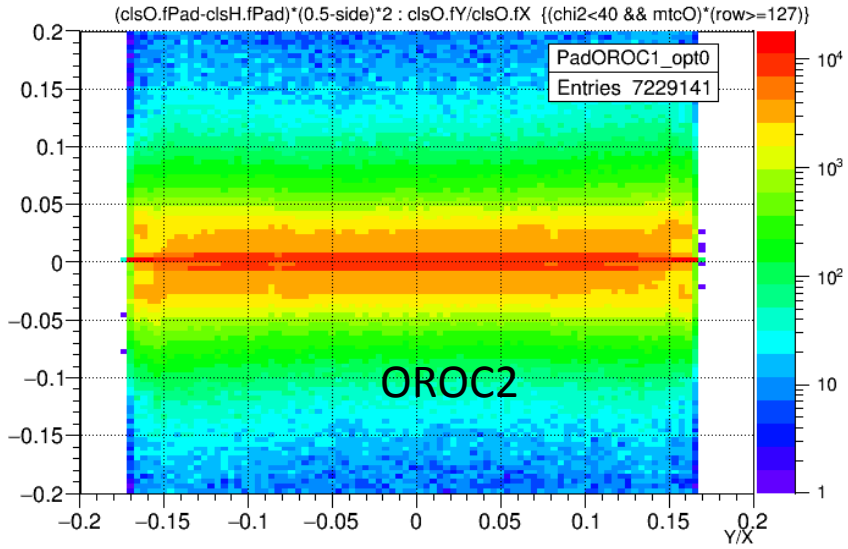


Pad coordinate difference: Offline - HLT

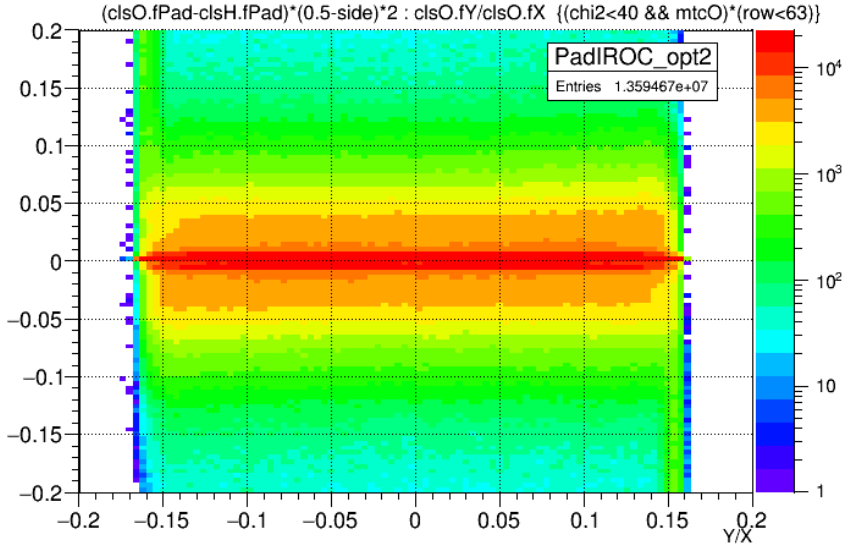
HLT = 0_ref (edge cluster position @ COG)

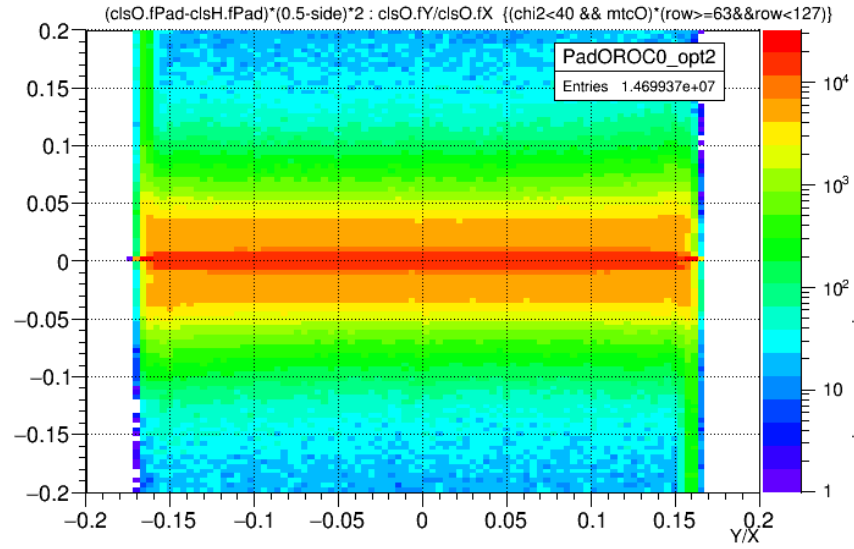


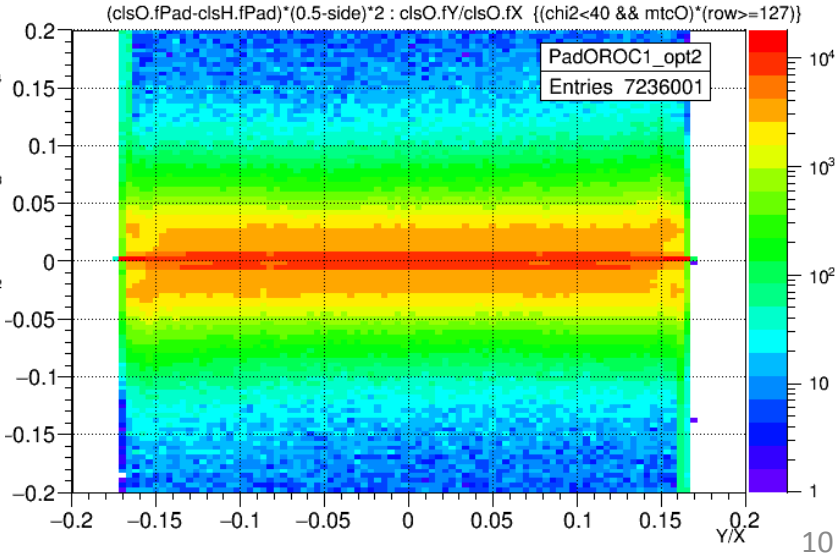




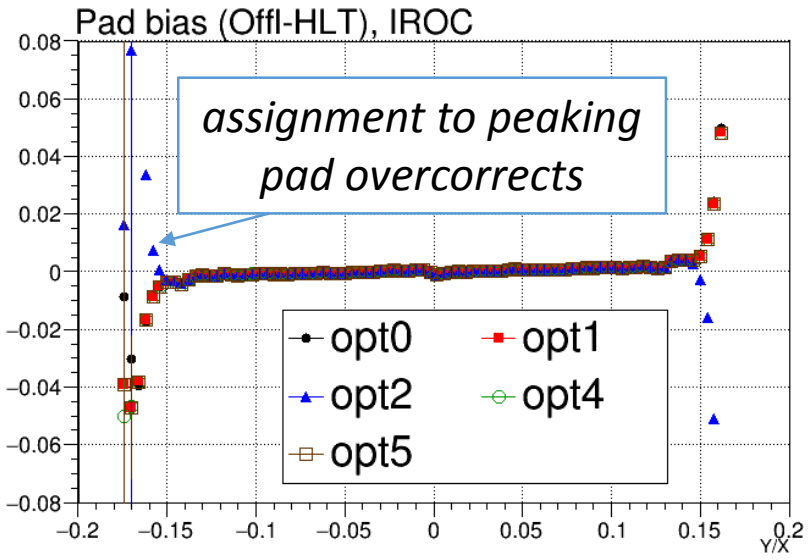
HLT = opt2 (edge cluster position at peaking pad)

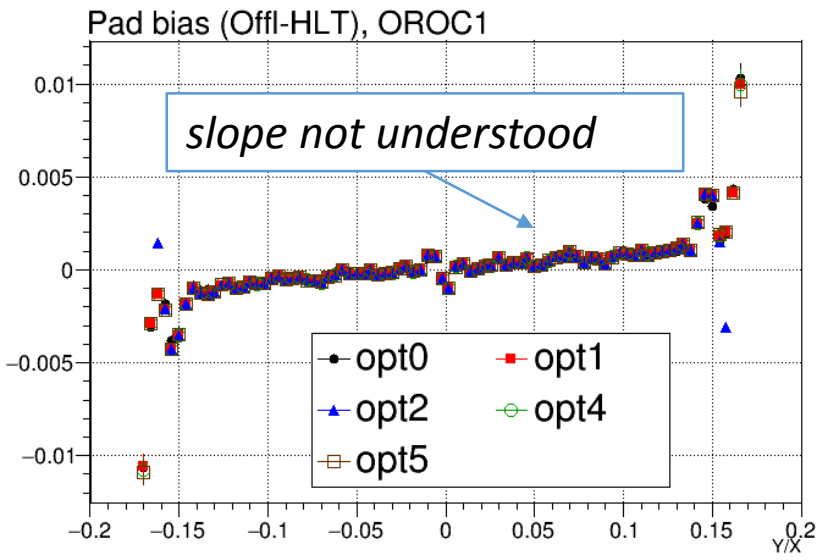


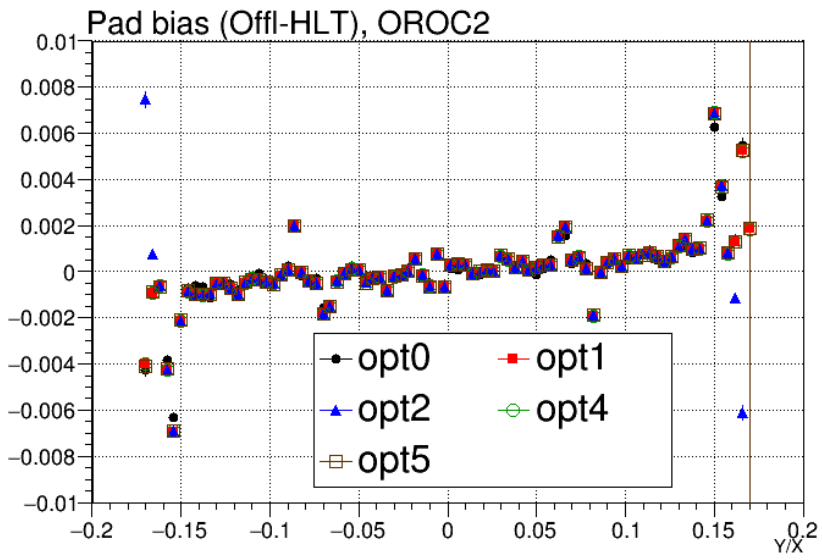




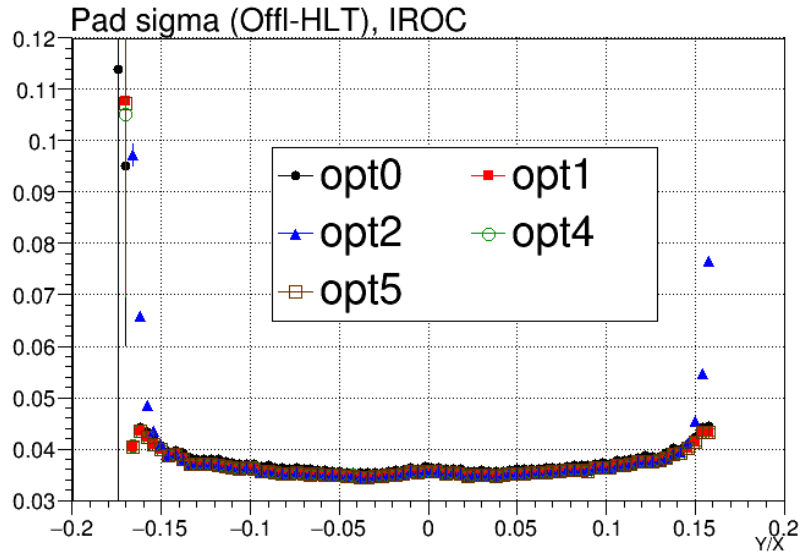
Mean and sigma of pad coordinate differences

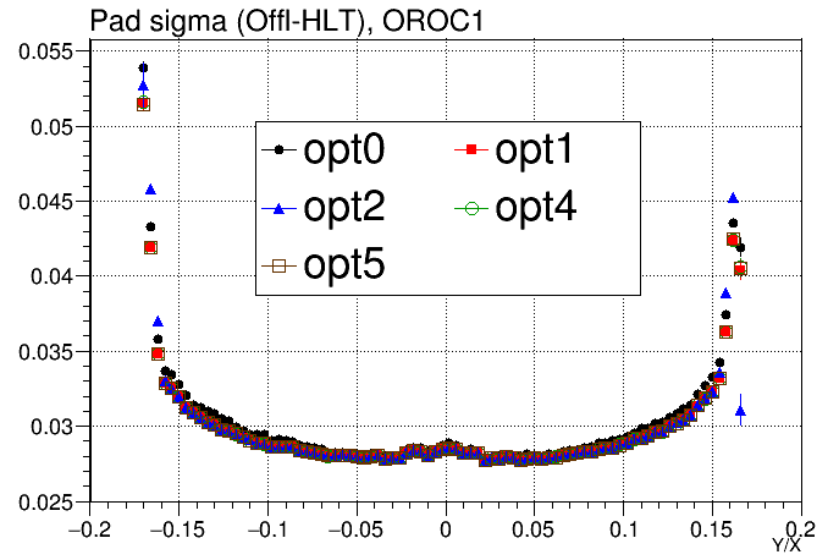


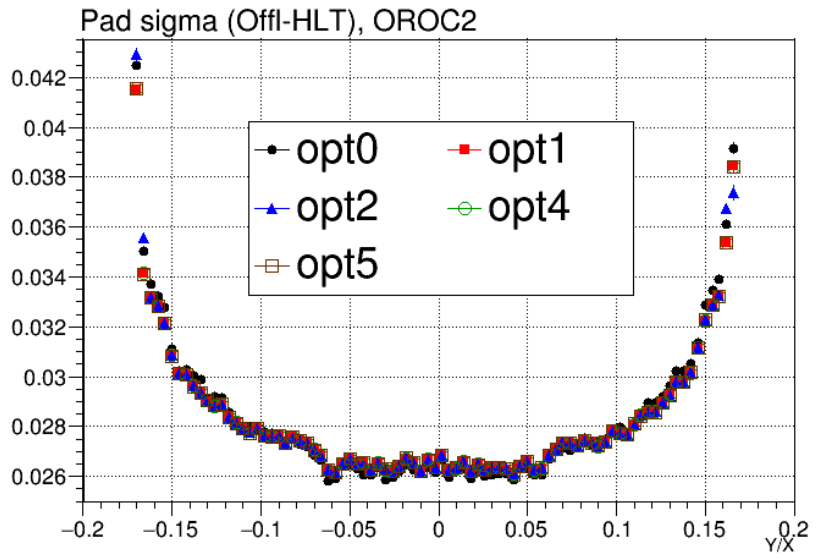




Opt2 “improvement” overcorrects the edge bias







MC to MC embedding

MC to MC embedding (sim,rec)

- Idea (see presentation by Francesco for details)
  - Simulate heavy underlying (PbPb) event up to the level of SDigits and use as background events library
  - Do “signal injected” MC productions by reading underlying events and adding to it signal generated with the same vertex and time stamp. Same underlying event can be reused multiple times.
- Framework for overlying SDigits and read MC data from multiple sources existed since long
  … but required multiple minor fixes:
  - Using time stamp of underlying event for signal one: critical for detectors with time-aware calibrations (e.g. TPC)
  - MC labels building in FMD, VZERO, AD, TOF, TRD
  - ITS needs MC truth in clusterization of MC data (related to fixed amount of labels we store per cluster): to reconstruct embedded MC data one should provide to AliReconstruction an AliMCEventHandler configured to read both underlying and signal events.
  - AliMCEventHandler was assuming 1 to 1 correspondence between bg. and signal event IDs (now replication factor is stored in galice.root and MC header (TE) trees.
- Current status: the framework is ready on AliRoot level (simulation + reconstruction), need validation by DPG

MC to MC embedding (analysis)

In ESD->AOD filtering MC truth from different sources is merged to standard single AOD MC info
⇒ no modifications train/tasks are needed for AOD analysis (when we manage to produce AOD…)

Analysis trains requiring ESD input need to be configured with AliMCEventHandler accessing both background and signal simulations:

```
AliAnalysisManager* mgr = AliAnalysisManager::GetAnalysisManager();
...
// usual handler we create, reads from analysed production directory
AliMCEventHandler* mcHandler = new AliMCEventHandler();
mgr->SetMCtruthEventHandler(mcHandler);
mcHandler->SetPreReadMode(1);
mcHandler->SetReadTR(kFALSE);
TString bgDir = gSystem->Getenv("CONFIG_BGEVDIR"); // defined only for embedded MC
if (!bgDir.IsNull()) { // add extra handler for underlaying event
  if (!bgDir.EndsWith("/")) bgDir += "/";
  // extra handler to read underlying event from CONFIG_BGEVDIR directory
  AliMCEventHandler* mcHandlerBg = new AliMCEventHandler();
  mcHandlerBg->SetInputPath(bgDir.Data());
  mcHandler->AddSubsidiaryHandler(mcHandlerBg);
  mcHandlerBg->SetPreReadMode(1);
  mcHandlerBg->SetReadTR(kFALSE);
}
```

MC to MC embedding (analysis)

Major problem in AliPhysics: currently most of tasks accessing ESD MC truth as:

```
AliMCEvent* fMCEvent = MCEvent();
AliStack* fMCStack = fMCEvent->Stack();
…
Tparticle* part = fMCStack->Particle(label); // will crash with embedded MC
// when accessing label of bg. event: Stack knows only signal event
```

Embedded MC truth in ESD analysis should be accessed only via AliMCEvent!
Due to this we currently cannot run neither AODtrainsim.C nor Qatrain_sim.C

Massive patching of AliPhysics tasks by PWGs will be needed:

	AliStack	AliMCEvent

`Tparticle* p=`	`stack->Particle(id);`	`((AliMCParticle*)mcevent->GetTrack(id))->Particle();`
		or
		`mcevent->Particle(id); // needs new AliRoot`

`int n=`	`stack->GetNtrack();`	`mcevent->GetNumberOfTracks();`
`int n=`	`stack->GetNprimary();`	`int n = mcevent->GetNumberOfPrimaries();`

`bool v=`	`stack->IsPhysicalPrimary(id);`
	`IsSecondaryFromWeakDecay(id);`
	`IsSecondaryFromMaterial(id)`
		`bool v= mcevent->IsPhysicalPrimary(id);`
		`IsSecondaryFromWeakDecay(id);`
		`IsSecondaryFromMaterial(id);`

VDT wrapper for math calculations

VDT-wrapper

- **Fact:** ~15% of CPU time in simulation spend in standard math function calls, ~6% in reconstruction

- **Objective:** Study ways to reduce this time by using a different math library implementation; Make this agnostic to the user — ideally without code change in AliRoot; Study potential gain and impact on physics observables

- **Idea:** Use vdt library (part of ROOT; github.com/dpiparo/vdt)
  - fast implementations based on Cephes library; similar to what is offered in Intel MKL
  - small compromise on precision; reduced error handling; reduced range
  - see vdt paper; see Sandro's talk in offline week

- **Things done:** Implemented a **vdt-wrapper**
  - can "intercept" calls to libm
  - dispatch to vdt when appropriate
  - dispatch to libm in corner cases or when out of range
  - "`LD_PRELOAD=alivdtwrapper.so aliroot sim.C`" is all there is to do …

S.Wenzel

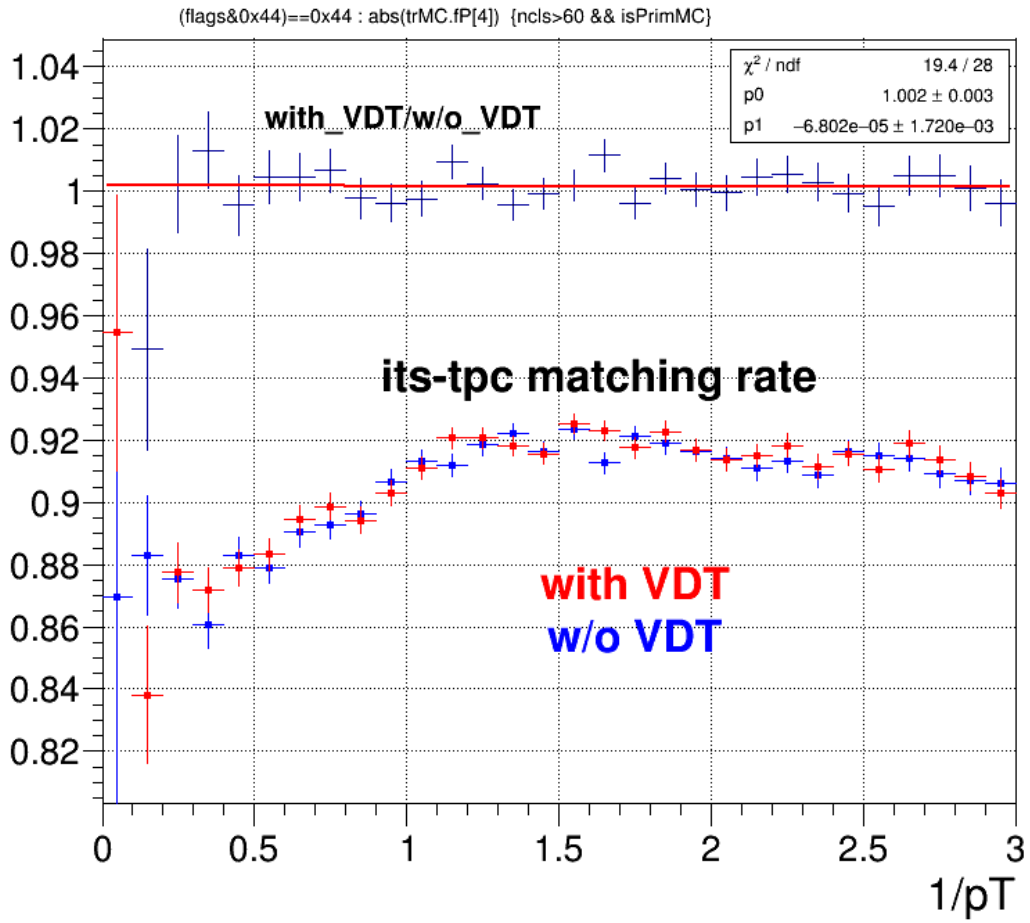
Timing test

same external primary MC input simulated and reconstructed with and w/o VDT wrapper

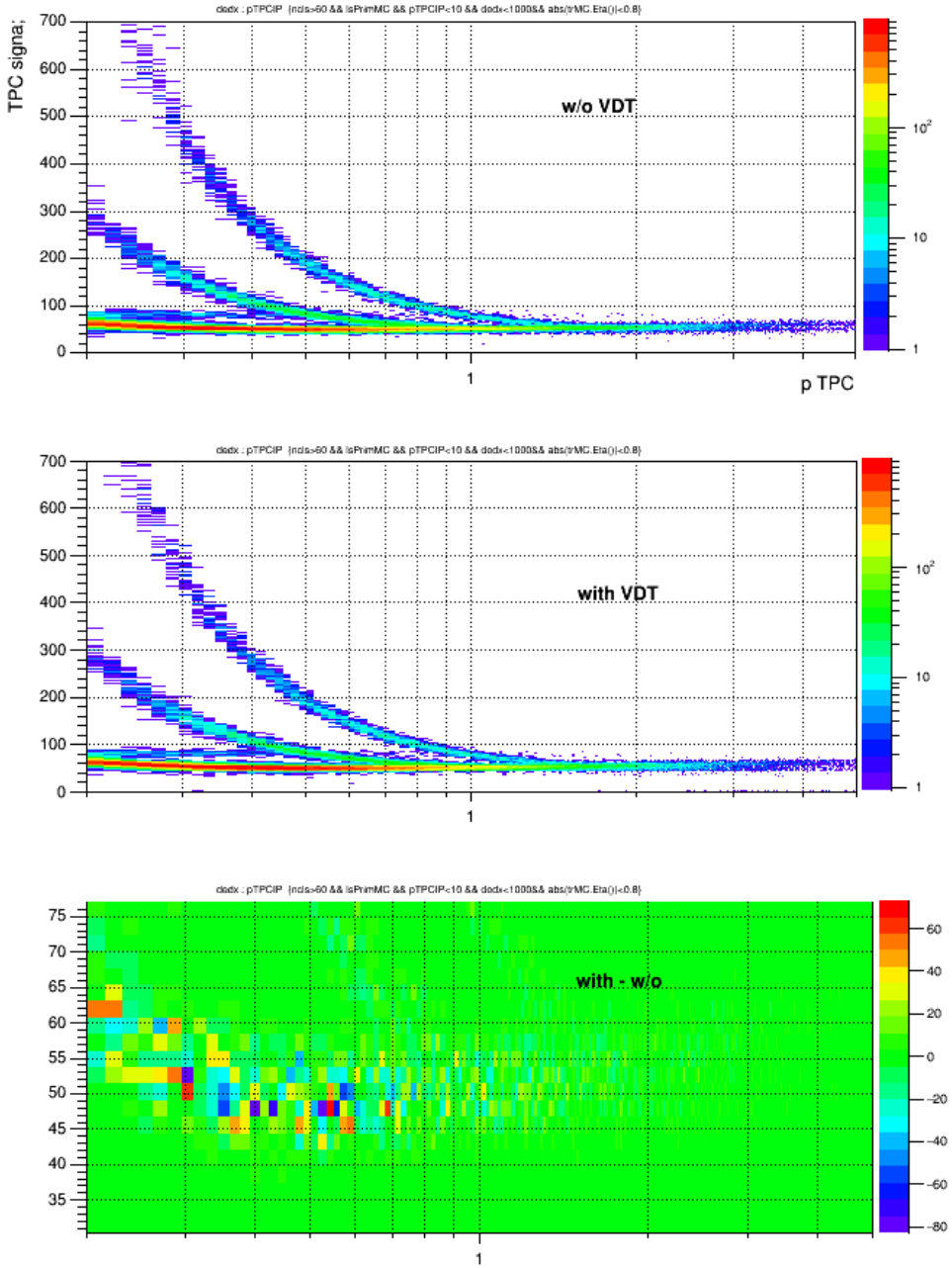
	w/o VDT	with VDT	gain, %
GRID (200 jobs of 100 pp events)			
Simulation (s/event)	79.4	74.3	**6.4**
Reconstruction (s/event)	2.17	2.07	**4.6**
LXPLUS (50 pp events)			
Simulation (s/event)	67.1	65.7	**2.0**
Reconstruction (s/event)	1.66	1.6	**3.4**

S.Wenzel, R.S

Reconstruction performance test



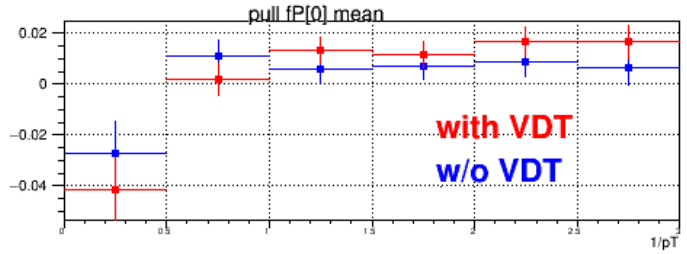
TPC dEdx vs p

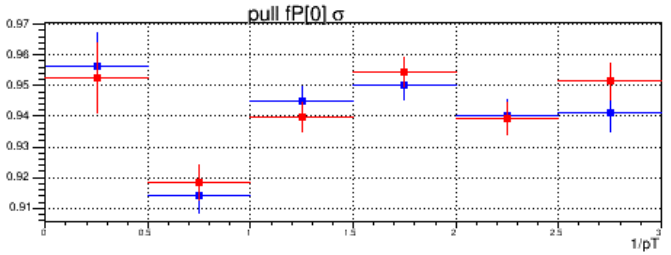


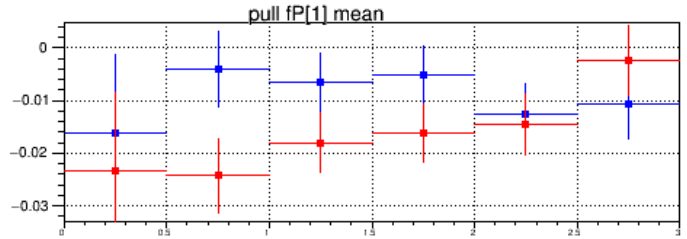
Pulls of track parameters at the vertex vs MC truth

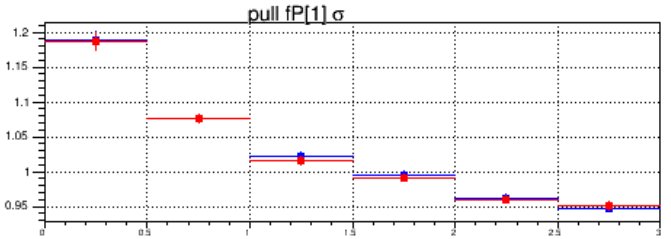
⇓

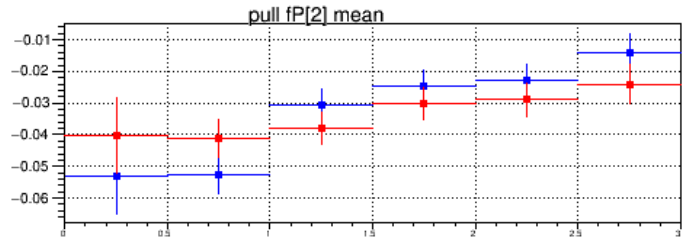
No systematic differences

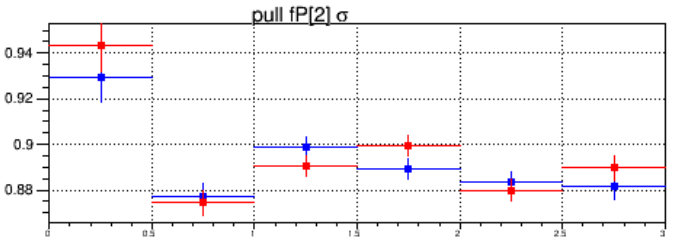


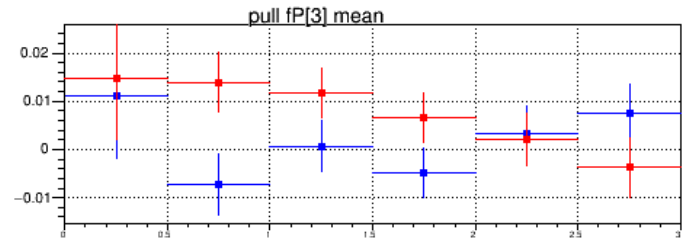


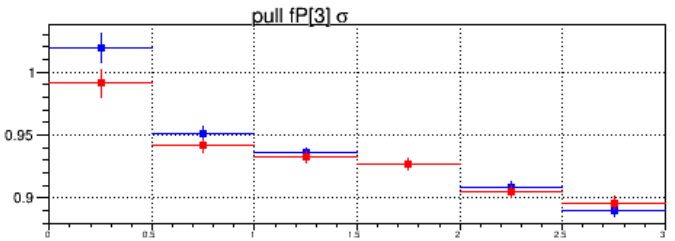


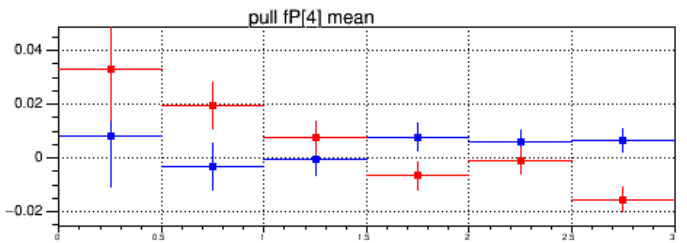


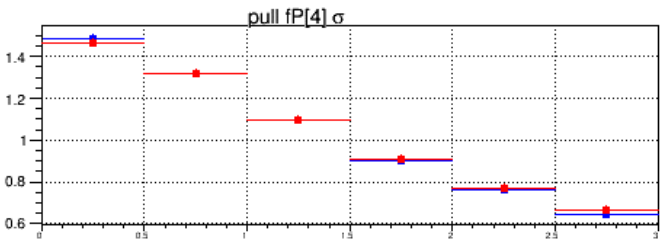












Reconstruction with TRD contributing to track fit

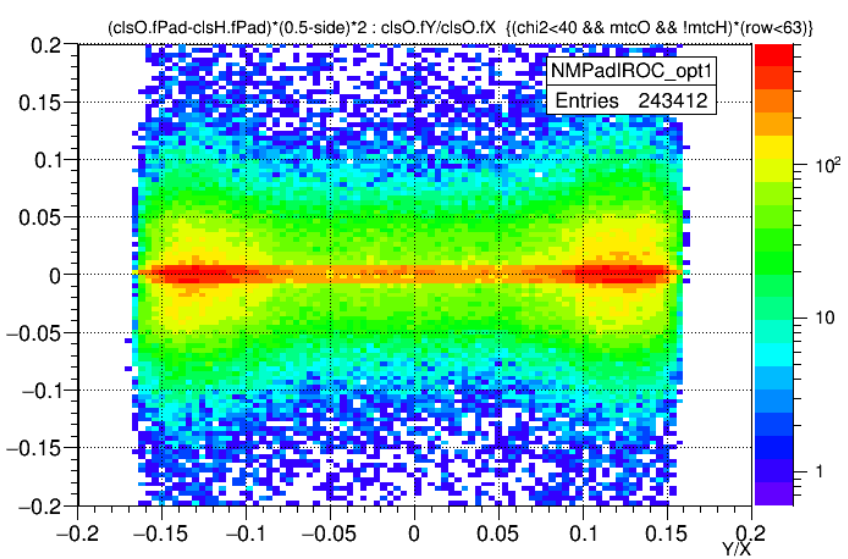
See following presentation by Marian

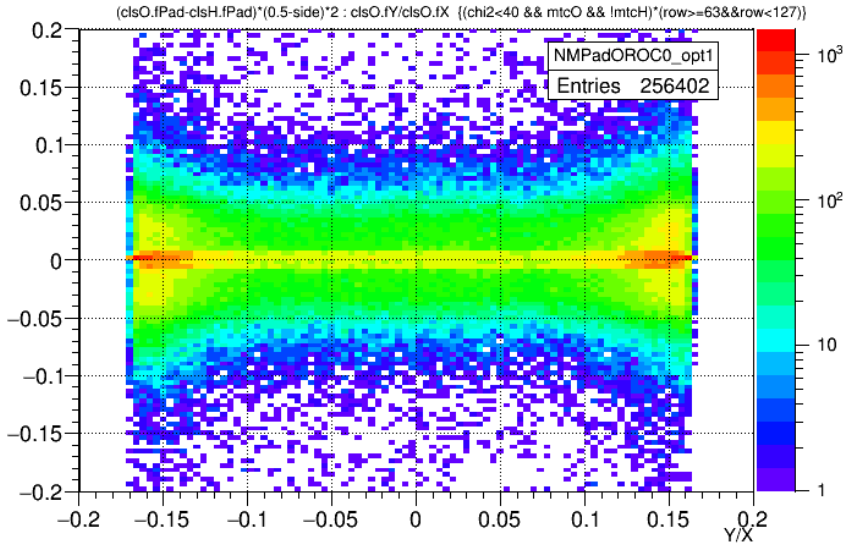
- MC confirms ~40% improvement in p_T resolution due to larger lever arm.
- Failed to achieve it with Run1 data in 2014 due to the residual TPC miscalibrations and defects of global alignment
- With improved alignment and new TPC SP calibration procedure hope to materialize expected improvement (at least at moderate IR) and compensate for space-charge distortion fluctuation effects at high IR.

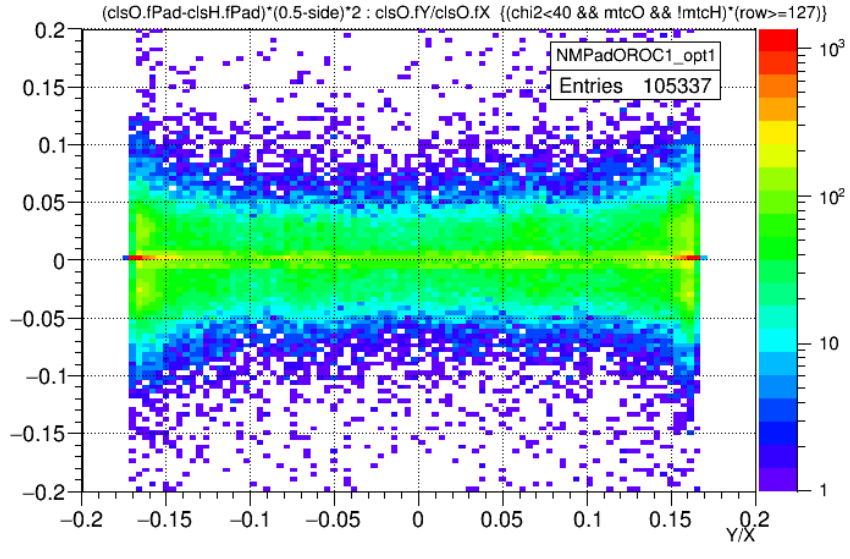
Faster parametrization for B-field

- Currently ~5% of CPU time in offline reconstruction is spent on B-field queries
  ⇒ will be too expensive in O2
- HLT uses simple Pol2 parametrization for Bz component only (~1% error): part of the reason of slightly worse HLT p_T resolution wrt offline (still OK for track-finding stage)

Shuto Yamasaki working on building faster parametrization for 3D field
(with main focus on O2 but hope to profit also in Run2)







Pad coordinate differences (Offl-HLT) for track NOT matched with HLT clusters but matched with offline clusters