KF / GBL comparison on tri-trig MC and Data 2019

- Analysis Workshop 2019 data -

PF, RJ, TT

06/04/2020

Outline of KF / GBL checks

- Apart from processing time I tried to make a general comparison between events reconstructed with KF and GBL Tracks
- Today:
  - **Using 2016 MC trident+beam sample:**

    - Check on track parameter pulls using truth matching (similar to what Robert has already shown)
  - **Using 2019 MC trident+beam sample:**

    - Present the current configuration of the Kalman Pat Recognition

    - Number of tracks reconstructed per event and extrapolation to ECAL
  - **First look at 2019 Data:**

    - Used 10031 to get a feeling of current detector performance

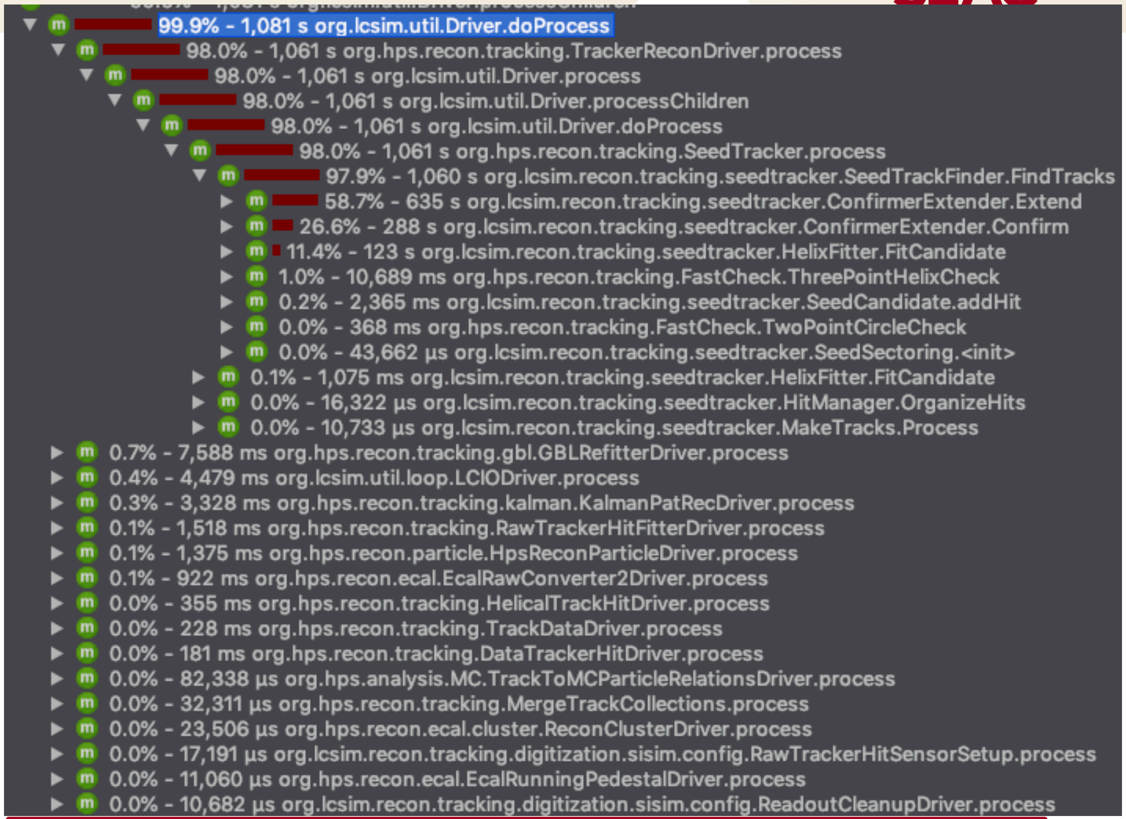
    - Vertex resolution as function of VtxP and Vtx InvMass compared to MC simulation

    - KF "Unbiased Residuals" per layer
  - **Summary and to-do**

Processing time - Tri-Trig ***with Beam***

SLAC

- File tested: /nfs/slac/g/hps3/users/bravo/mc/mc2019/tritrig/readoutFromJLAb/tritrig_1.slcio
- With the current strategies, tracking takes:
  - **~98% in the SeedTracker** (60% in the extension, 27% in the confirmation, 12% in the fitting)
  - Mostly due to very large cuts in rmsTime in SeedTracker (1000ns), but also setting it at (20ns) doesn't help (98% => 93% see Backup)
  - ~0.7% in GBL Refitting stage
- Kalman track finding and fitting takes ~0.3% of the event time in this conditions
- All the rest of the event reconstruction time becomes negligible
- **Not sustainable for high-stat MC or reReco passes.**
- **Total time: 25m for ~150 events on cent7a => 10s/event**



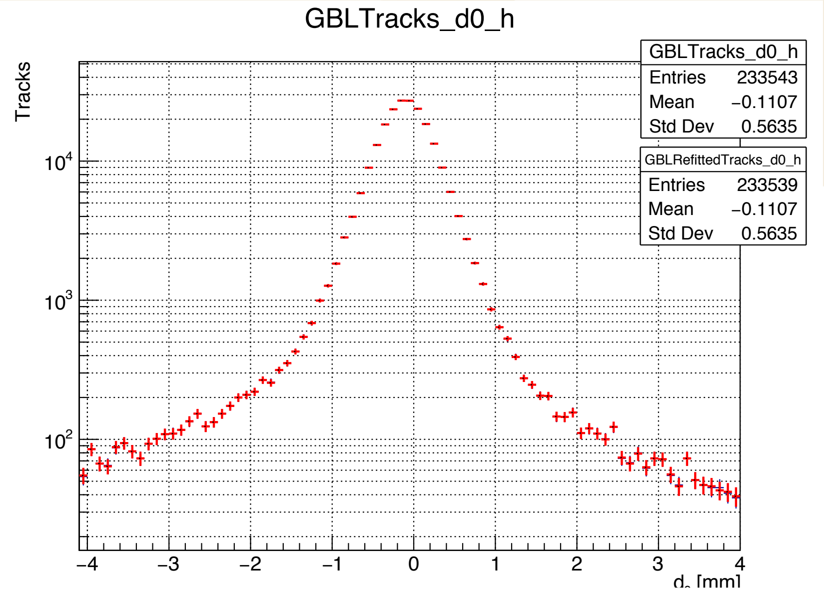
Total Tracking time ~98% in tri-trig signal with beam background. Not sustainable in long run. A more detailed dump in the backup

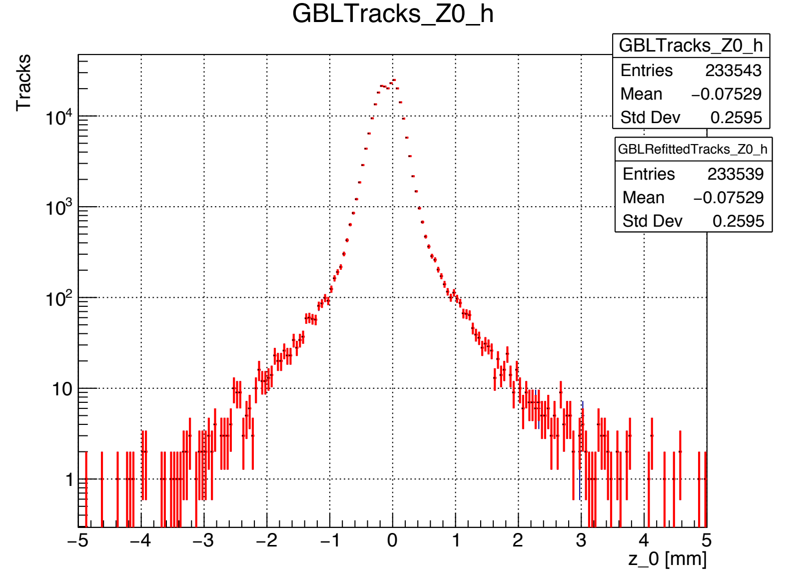
jProfiler

Evaluation version, remotely attached to cent7a, **readout to LCIO step**

Refit GBLTracks with KF

- First check was to refit the pre-reconstructed GBL tracks using Kalman Filter routines
- In "official" reconstructed 2016 trident+beam MC samples TSOS are not stored
  - Necessary for refitting with KF (as it needs a seed for the first state)
  - Procedure
    - Refit GBLTracks from Matched tracks => **GBLRefittedTracks**
    - Checked that GBLTracks (original) and GBLRefittedTracks (refitted) have same track parameters - negligible differences
    - Store TSOS
    - After confirming that, proceeded to refit using KF





Hit Content Check

```
collection name : KalmanFullTracks
parameters:

--------------- print out of Track collection ---------------

  flag:  0x80000000
     LCIO::TRBIT_HITS : 1

 [   id   ] |   type   |    d0    |  phi     | omega    |    z0     | tan lambda|   reference point(x,y,z)        |    dEdx  |  dEdxErr |   chi2   |  nd\
f
------------|----------|----------|----------|----------|-----------|-----------|---------------------------------|----------|----------|--------
 [00000567] | 00000001 |+7.17e-01 |+2.03e-02 |-3.13e-04 |-9.685e-02 |+2.068e-02 |(+0.00e+00, +0.00e+00, +0.00e+00)|+4.76e-05 |+0.00e+00 |+8.48e+00 |     \
0
 errors:   +1.016003e-01 | -7.924382e-04, +7.207056e-06 | -1.320098e-06, +1.361114e-08, +7.121702e-11 | +1.466841e-02, -7.706471e-05, -1.437735e-07, +2.\
180240e-02 | -1.725753e-04, +9.061276e-07, +1.690655e-09, -2.467633e-04, +2.797374e-06 |
 tracks(id):
 hits ->[0000+605] [0000+599] [0000+603] [0000+645] [0000+613] [0000+614] [0000+646] [0000+607] [0000+633] [0000+643]
 radius of innermost hit +2.792673e+00 / mm ,  subdetector Hit numbers : +0
------------|----------|----------|----------|----------|-----------|-----------|---------------------------------|----------|----------|--------
```

```
collection name : GBLTracks
parameters:

------------- print out of Track collection ---------------

 flag:  0x80000000
    LCIO::TRBIT_HITS : 1

[   id   ] |   type   |    d0    |  phi     | omega    |    z0     | tan lambda|   reference point(x,y,z)        |    dEdx  |  dEdxErr |   chi2   |

-----------|----------|----------|----------|----------|-----------|-----------|---------------------------------|----------|----------|--------
[00000906] | 00000057 |+8.25e-01 |+1.78e-02 |-3.13e-04 |-9.[illegible]55e-02 |+2.066e-02 |(+0.00e+00, +0.00e+00, +0.00e+00)|+4.76e-05 |+0.00e+00 |+7.86e+00 |

errors:   +1.004819e-01 | -7.822804e-04, +7.101278e-06 | -1.225804e-06, +1.260442e-08, +6.353884e-11 | +1.459154e-02, -7.704410e-05, -1.381959e-07,
77905e-02 | -1.722984e-04, +9.104180e-07, +1.633262e-09, -2.466932e-04, +2.798859e-06 |
tracks(id):
hits ->[0000+657] [0000+659] [0000+661] [0000+662] [0000+663]
radius of innermost hit +9.209605e+01 / mm ,  subdetector Hit numbers : +0
```

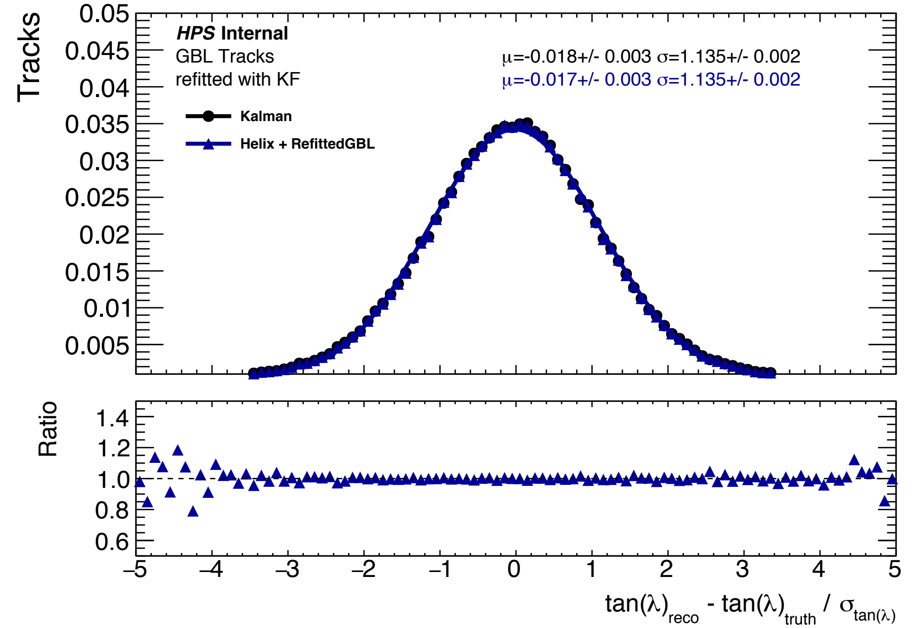
Same hits are picked
Can compare 1-to-1

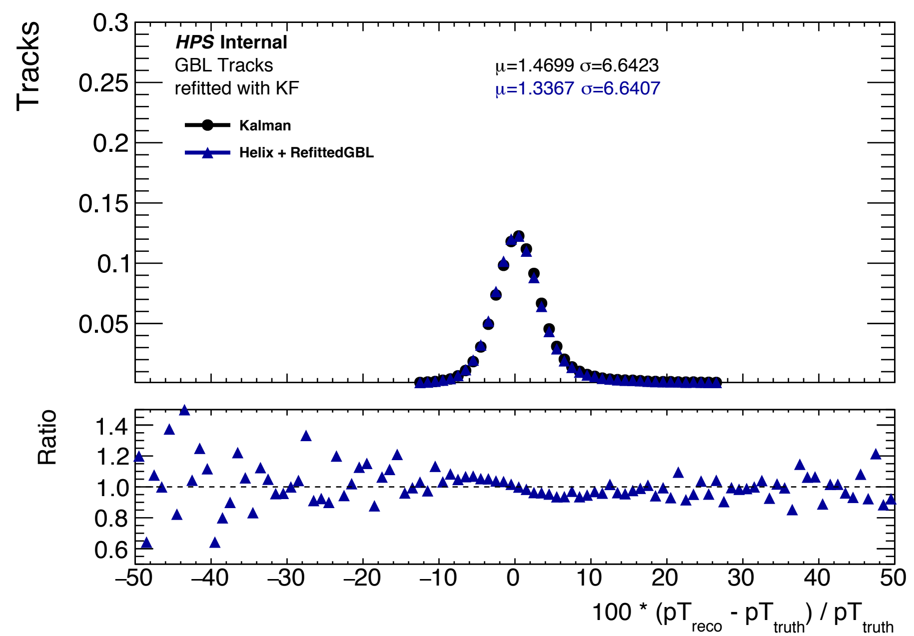
```
--------------- print out of LCRelation collection ---------------

 flag:  0x0
fromType :
toType :

| [from_id ] | [ to_id  ] | Weight   |
|------------|------------|----------|
| [0000067[illegible] | [00000605] | 1.000000|
| [00000677] | [00000599] | 1.000000|
| [00000678] | [00000649] | 1.000000|
| [00000678] | [00000639] | 1.000000|
| [00000679] | [00000603] | 1.000000|
| [00000679] | [00000645] | 1.000000|
| [00000680] | [00000629] | 1.000000|
| [00000680] | [00000621] | 1.000000|
| [00000681] | [00000613] | 1.000000|
| [00000681] | [00000614] | 1.000000|
| [00000682] | [00000646] | 1.000000|
| [00000682] | [00000607] | 1.000000|
| [00000683] | [00000633] | 1.000000|
| [00000683] | [00000643] | 1.000000|
```

```
collection name : RotatedHelicalTrackHitRelations
parameters:

--------------- print out of LCRelation collection ---------------

 flag:  0x0
fromType :
toType :

| [from_id ] | [ to_id  ] | Weight   |
|------------|------------|----------|
| [00000677] | [00000657] | 1.000000|
| [00000678] | [00000658] | 1.000000|
| [00000679] | [00000659] | 1.000000|
| [00000680] | [00000660] | 1.000000|
| [00000681] | [00000661] | 1.000000|
| [00000682] | [00000662] | 1.000000|
| [00000683] | [00000663] | 1.000000|
```

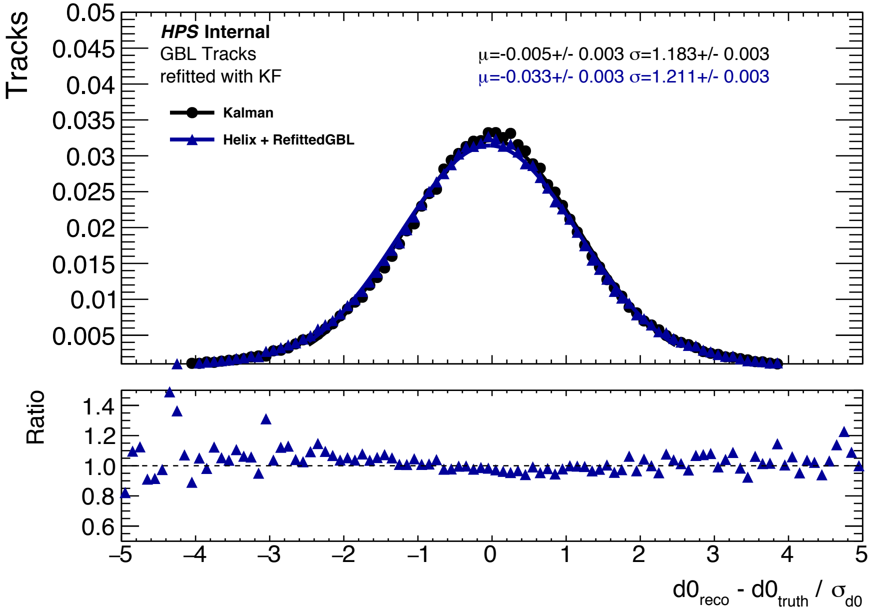
Truth matching and pull checks

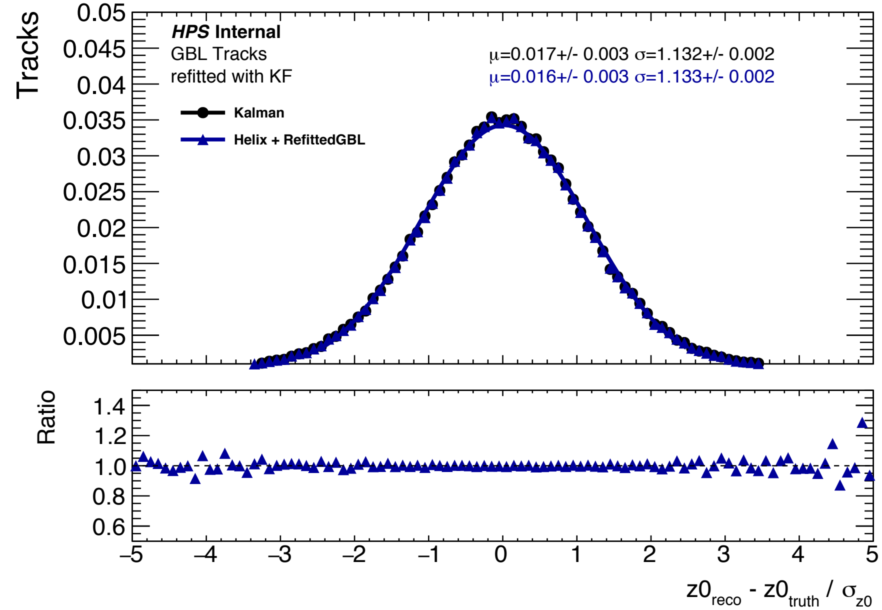
- Both KF tracks and GBL Tracks are matched to MCParticle (the matched particle is the one with highest # hits on track)
- Since KF is the GBL refit (using the same hits), they are matched to the same particle by definition
- Truth matching is done using TrackTruthMatching tool written by MattS
- MCParticle is then converted to HelicalTrackFit and then to LCIO::Event::Track to be persisted (see TrackToMCParticleRelationsDriver)
- Relations are kept, so can be exploited directly in hpstr in the future
- Momentum resolution only slightly worse wrt GBL, but at sub-percent level.

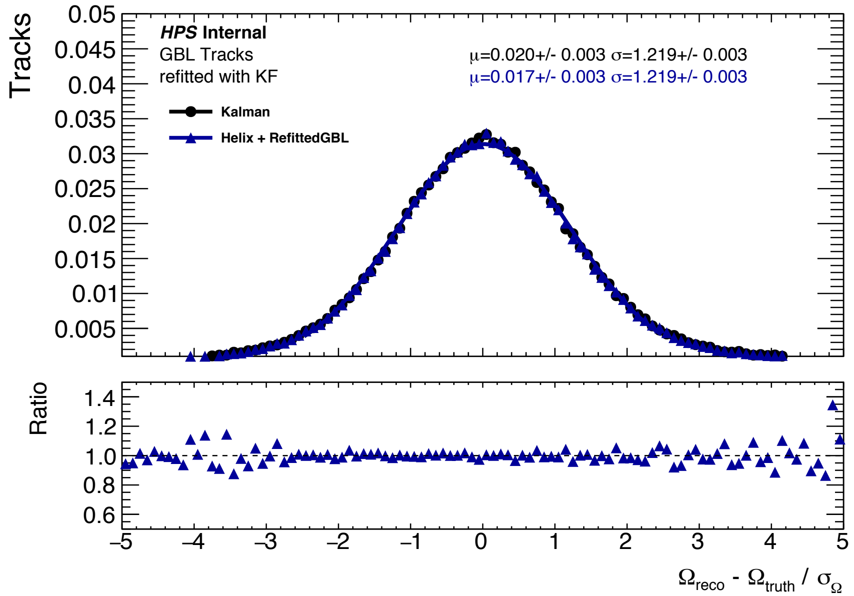


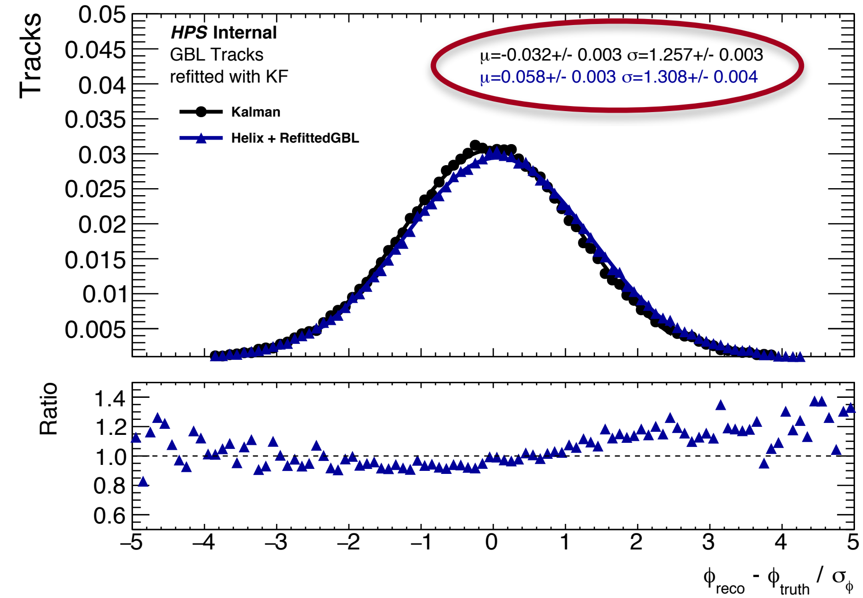


Truth matching and pull checks



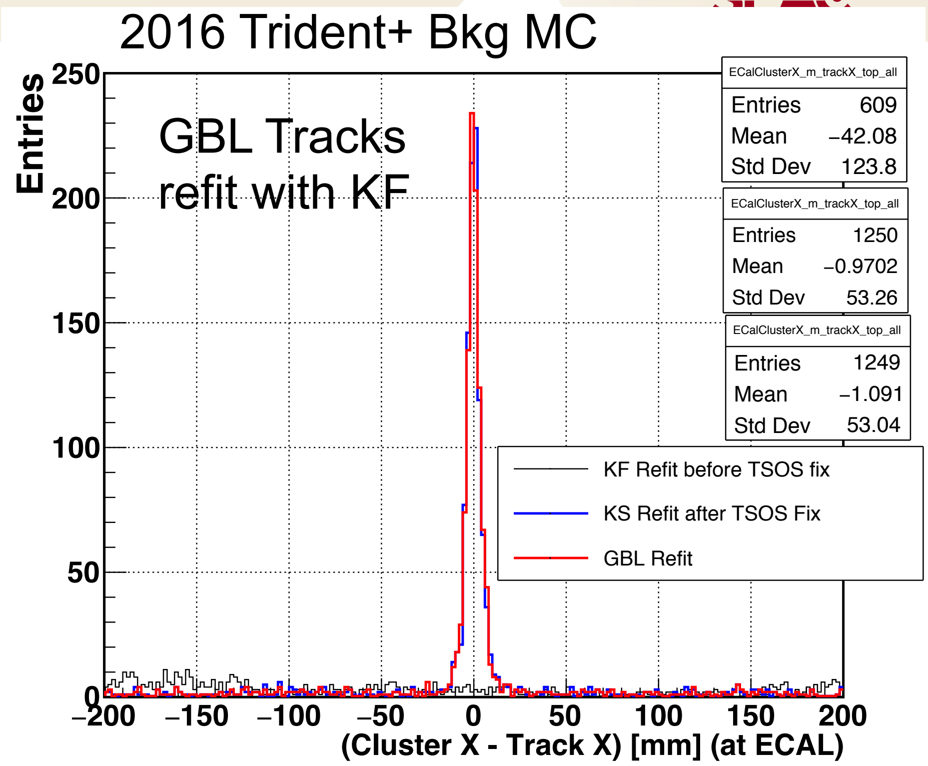






Extrapolation to Ecal for KF

- Minor changes to the Track States on surfaces to enable extrapolation to the ECAL in 2016 MC
- GBL Tracks refit with KF lead to same matching of ECAL clusters
- The extrapolation relies on previous RK method => Robert's new extrapolation should be checked



KF tracks are expected to have similar performance of GBL tracks when running on the same hit content. A sensible improvement in estimating d0 was observed.

Kalman Pat Reco / GBL in 2019 MC

SLAC

- Yesterday has been discussed that Kalman Pattern Reco has been enabled in the current MC reconstruction
- In the following slides, both SeedTracker and Kalman Pattern Reco are ran on exactly the same events at the same time so it's possible to compare the relative performance 1-to-1
- **However they follow different seed strategy, pattern recognition cuts and hit content**
- **The results will fold together the different track finding and fitting algorithms.**

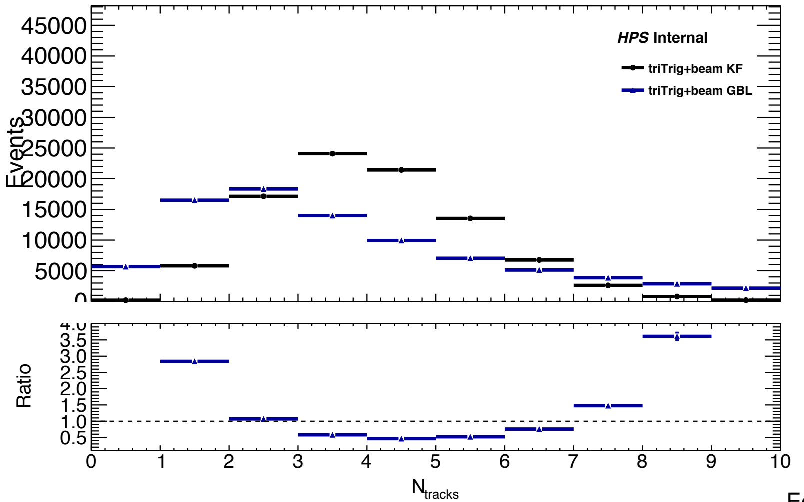
- An overview on how to setup KF in a reconstruction job is given in yesterday's talk

Kalman Pat Recognition tunable parameters

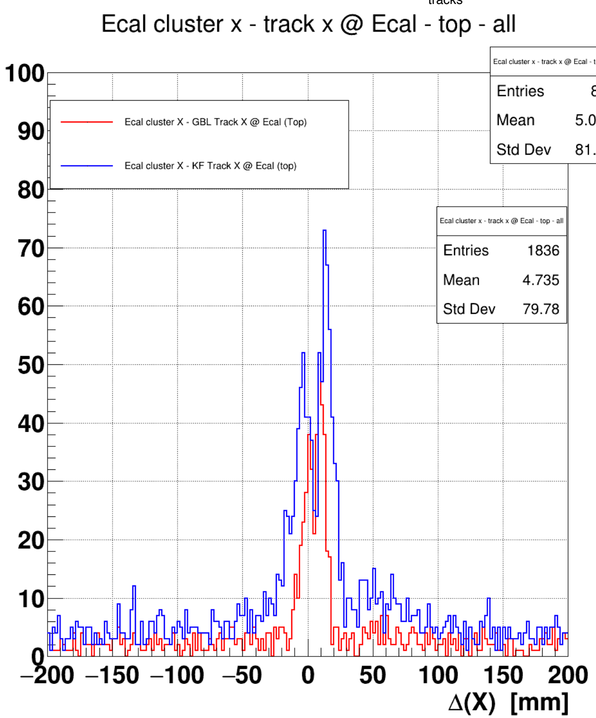
```
nIterations = 1;     // Number of Kalman filter iterations per track in the final fit
kMax[0] = 3.0;       // Maximum curvature for seed
kMax[1] = 6.0;
tanlMax[0] = 0.08;   // Maximum tan(lambda) for seed
tanlMax[1] = 0.12;
dRhoMax[0] = 15.;    // Maximum dRho at target plane for seed
dRhoMax[1] = 25.;
dzMax[0] = 3.;       // Maximum z at target plane for seed
dzMax[1] = 10.;
chi2mx1[0] = 8.0;    // Maximum chi**2/#hits for good track
chi2mx1[1] = 12.0;
minHits0 = 6;        // Minimum number of hits in the initial outward filtering (including 5 from the seed)
minHits1[0] = 7;     // Minimum number of hits for a good track
minHits1[1] = 6;
mxChi2Inc = 2.;  // Maximum increment to the chi^2 to add a hit to a completed track
minChi2IncBad = 10.; // Threshold for removing a bad hit from a track candidate
mxResid[0] = 50.;  // Maximum residual, in units of detector resolution, for picking up a hit
mxResid[1] = 100.;
mxResidShare = 10.; // Maximum residual, in units of detector resolution, for a hit to be shared
mxChi2double = 6.;  // Maximum chi^2 increment to keep a shared hit
minStereo[0] = 4;
minStereo[1] = 3;    // Minimum number of stereo hits
minAxial = 2;        // Minimum number of axial hits
mxShared = 2;        // Maximum number of shared hits
mxTdif = 30.;        // Maximum time difference of hits in a track
seedCompThr = -1;    // Remove SeedTracks with all Helix params within relative seedCompThr . If -1 do not apply duplicate removal
```

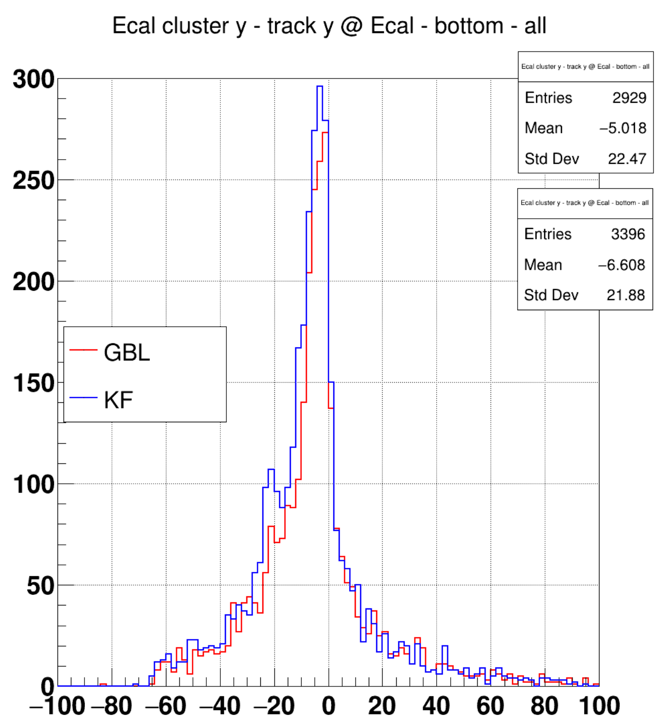
- Set of parameters is tunable from steering file
- Also list of seeding strategies configurable
- Hard to make a 1-to-1 comparison with the seeding strategies in SeedTracker

Number of tracks and extrapolation to ECAL in TriTrig+Beam 2019 MC



- **Tracks here** is the size of the Track container, i.e. the full GBLTracks and KF Tracks => **all that pass reconstruction** and no track/vtx/event selection is applied





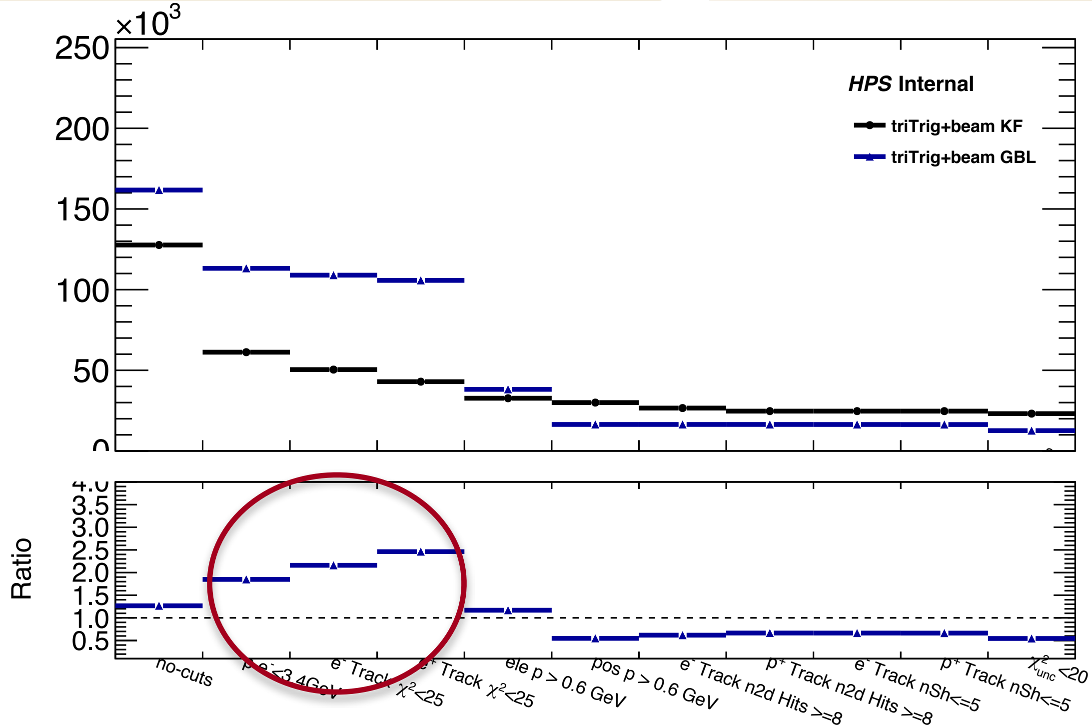
- KF **ECAL Extrapolation seems comparable to GBL**

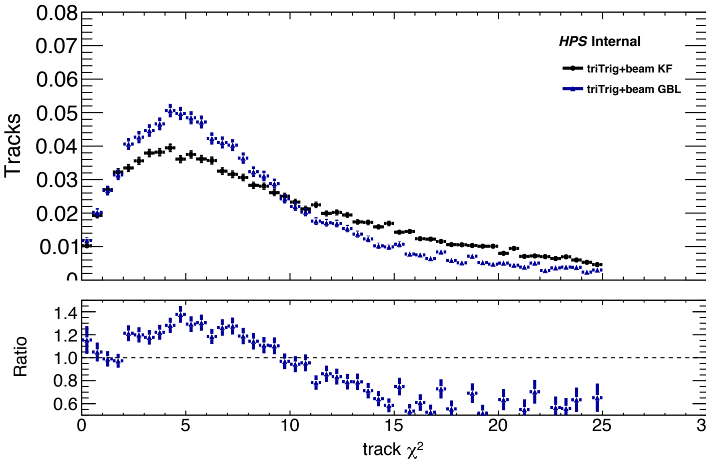
Basic checks on KF / GBL performance in MC 2019

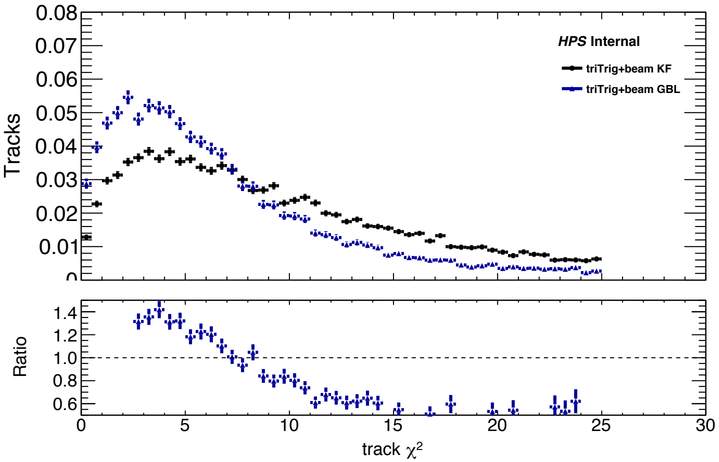
- I started checking KF vs GBL performance in events where full reconstruction is performed:
  - Both KF and GBL tracks are formed following their own pattern recognition
  - They are fed to the ReconParticleDriver to form vertices (constrained/ unconstrained [I only checked unconstrained so far])
- The data LCIO files can be found:
  - **/nfs/slac/g/hps3/users/pbutti/2019_data_10031/**
  - **/nfs/slac/g/hps3/users/pbutti/2019_tridents_from_LCIO_VtxFix**
- The processed hipster ntuples for analysis can be found at the same location:
  - **2019_data_10031_KFHitOnTracks**
  - **2019_tridents_from_LCIO_VtxFix_hpstr_ntuples**
- Only MOUSE cuts are applied to those ntuples.

Checks on MC - Basic cleaning cuts

- Check over V0 vertices
- I require:
  - e P < 3.4 GeV
  - e-/e+ Chi2 < 25
  - e-/e+ P > 0.6 GeV
  - 2D hits e-/e+ >= 8
  - e-/e+ NShared < 5 [no effect: MOUSE cuts]
  - Vtx Chi2 < 20



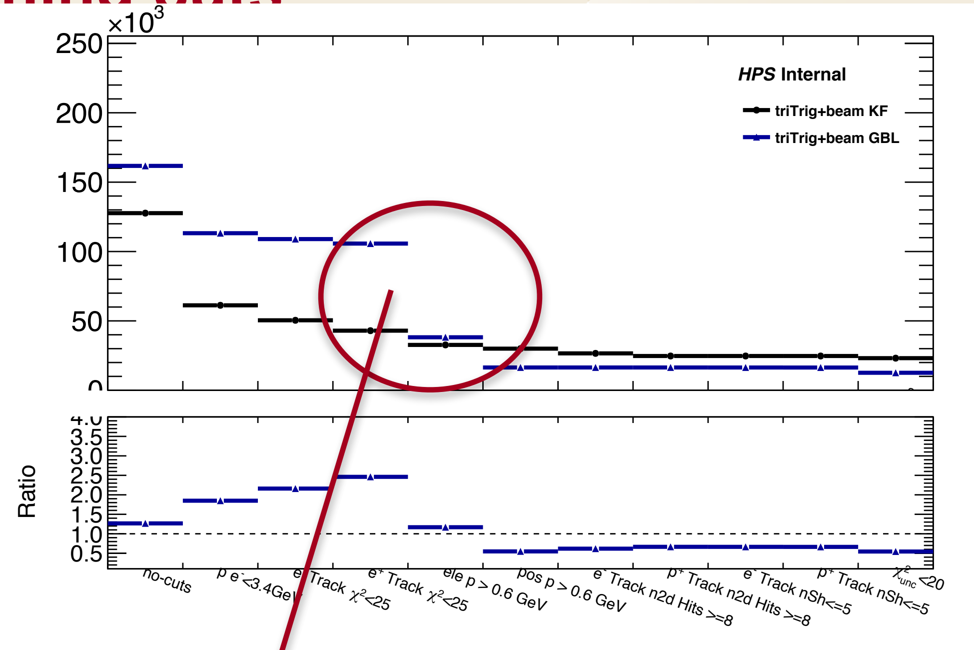


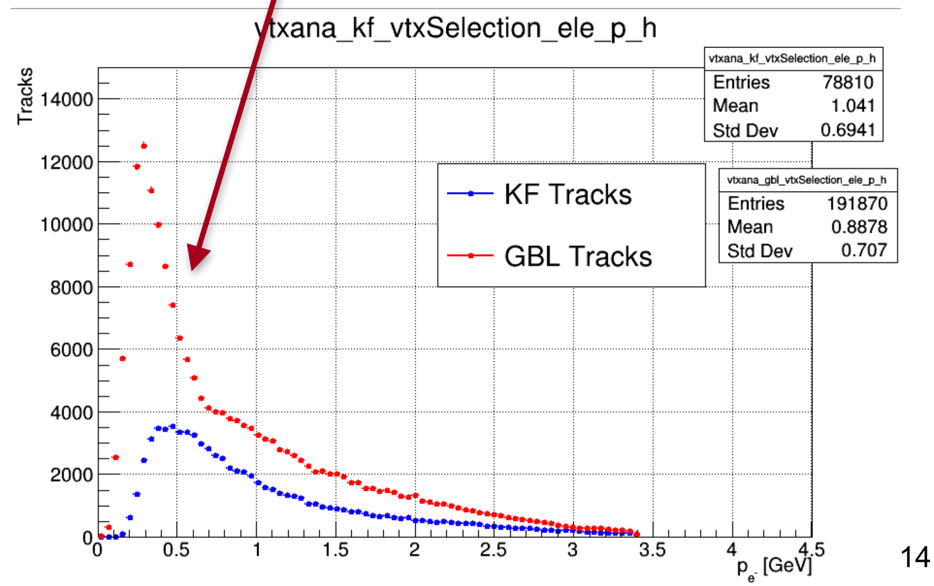


Larger track Chi2 in KF leads to eff drop wrt GBL. Not necessary we should trust GBL Chi2 value.

Checks on MC - Basic cleaning cuts

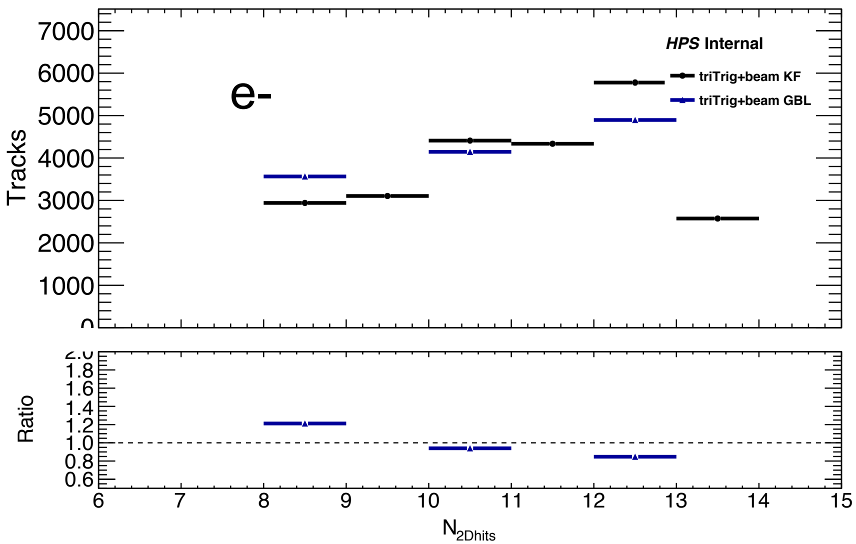
- Check over V0 vertices
- I require:
  - e P < 3.4 GeV
  - e-/e+ Chi2 < 25
  - e-/e+ P > 0.6 GeV
  - 2D hits e-/e+ >= 8
  - e-/e+ NShared < 5 [no effect: MOUSE cuts]
  - Vtx Chi2 < 20

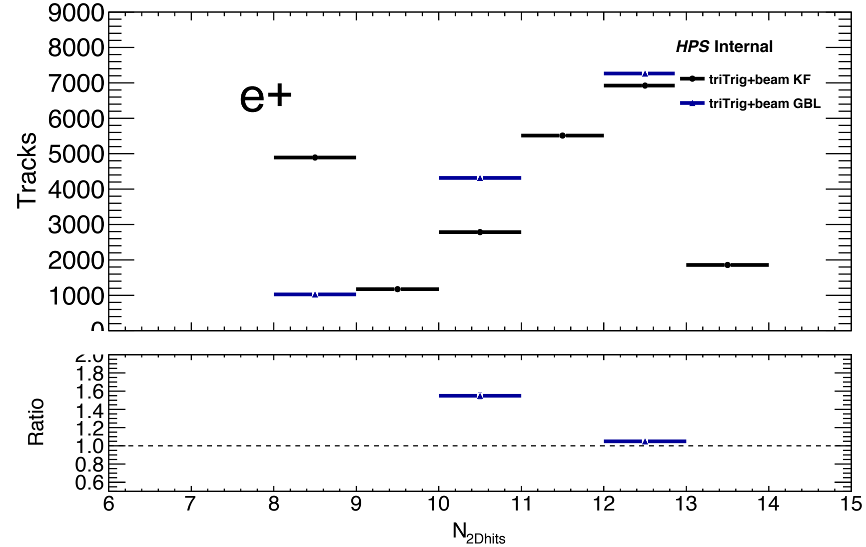


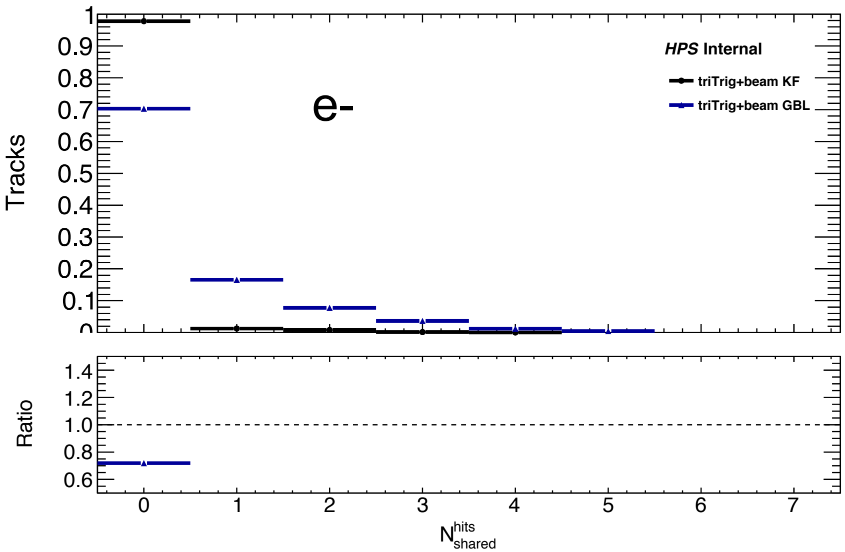


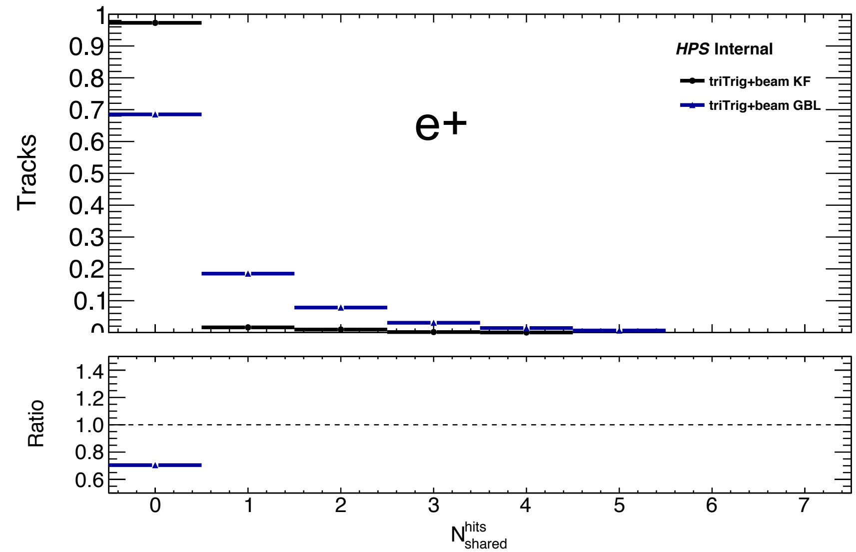
- When adding beam bkg to tri-trig signal, SeedTracker finds many more tracks at low momenta
- They contribute anyway to large UncVtx Chi2 and would be cut anyway by vertex quality requirements
- Kalman Pat Reco has a cut on pT > 0.3 (0.15) GeV in first (second) seeding stage pass

Hit Content


HPS Internal
triTrig+beam KF
triTrig+beam GBL
e-
Tracks
Ratio
N_{2Dhits}

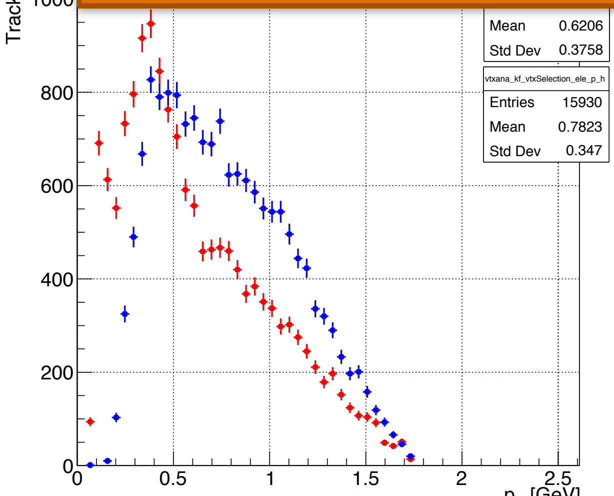

HPS Internal
triTrig+beam KF
triTrig+beam GBL
e+
Tracks
Ratio
N_{2Dhits}


HPS Internal
triTrig+beam KF
triTrig+beam GBL
e-
Tracks
Ratio
N^{hits}_{shared}

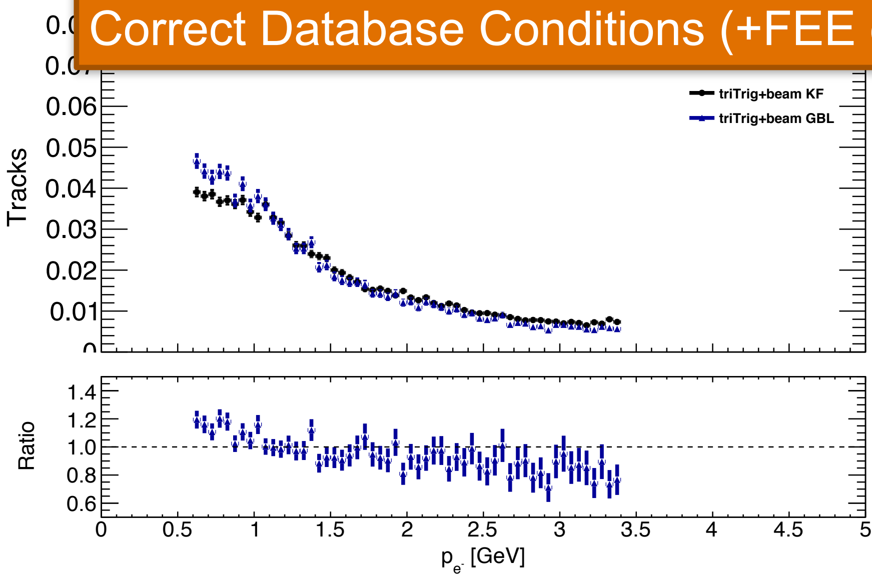

HPS Internal
triTrig+beam KF
triTrig+beam GBL
e+
Tracks
Ratio
N^{hits}_{shared}

Fixed momentum bug - Condition Database update

Wrong Database Conditions



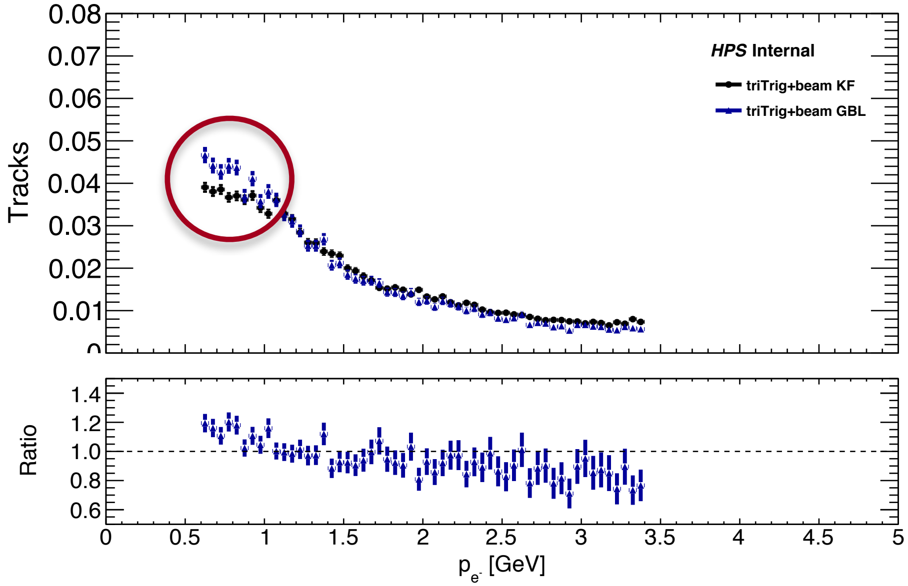
Correct Database Conditions (+FEE cut)

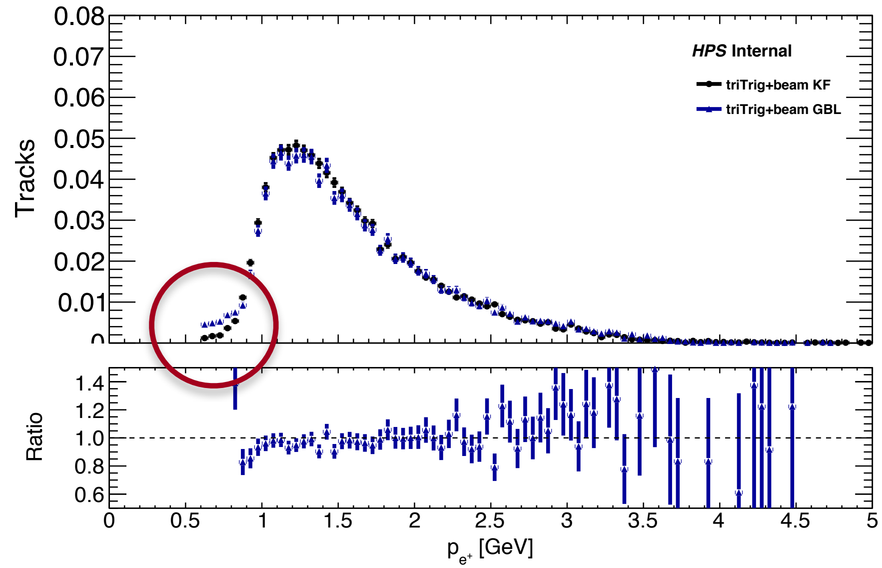


run_start	run_end	collection_id	notes	beam_energy
3000	3999	1988	engrun2015 beam energy	1.92
4000	6999	1987	engrun2015 beam energy	1.056
7000	8999	1973	nominal beam energy for physrun2016	2.306
9000	999999	3150	nominal beam energy for 2019 run	4.55
1000000	1000000	2993	beam energy for L0 MC	4.4
1000001	1000001	3142	beam energy at 6.6 GeV for L0 studies	6.6
1000011	1000011	3150	MC beam energy at 4.55 GeV.	4.55
2000000	2000000	3150	Beam energy for 2019 MC	4.55

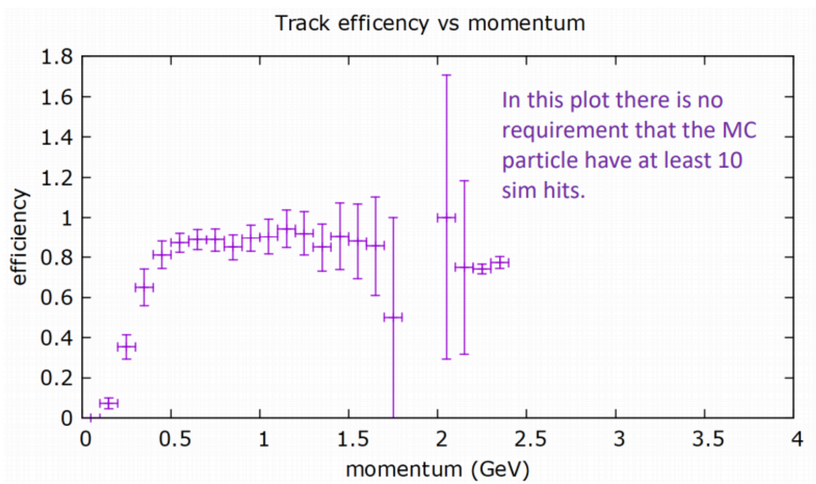
Beam Energy now fixed in database for 2019 MC/Data processing (thanks to Jeremy)

Electron/Positron momentum spectrum

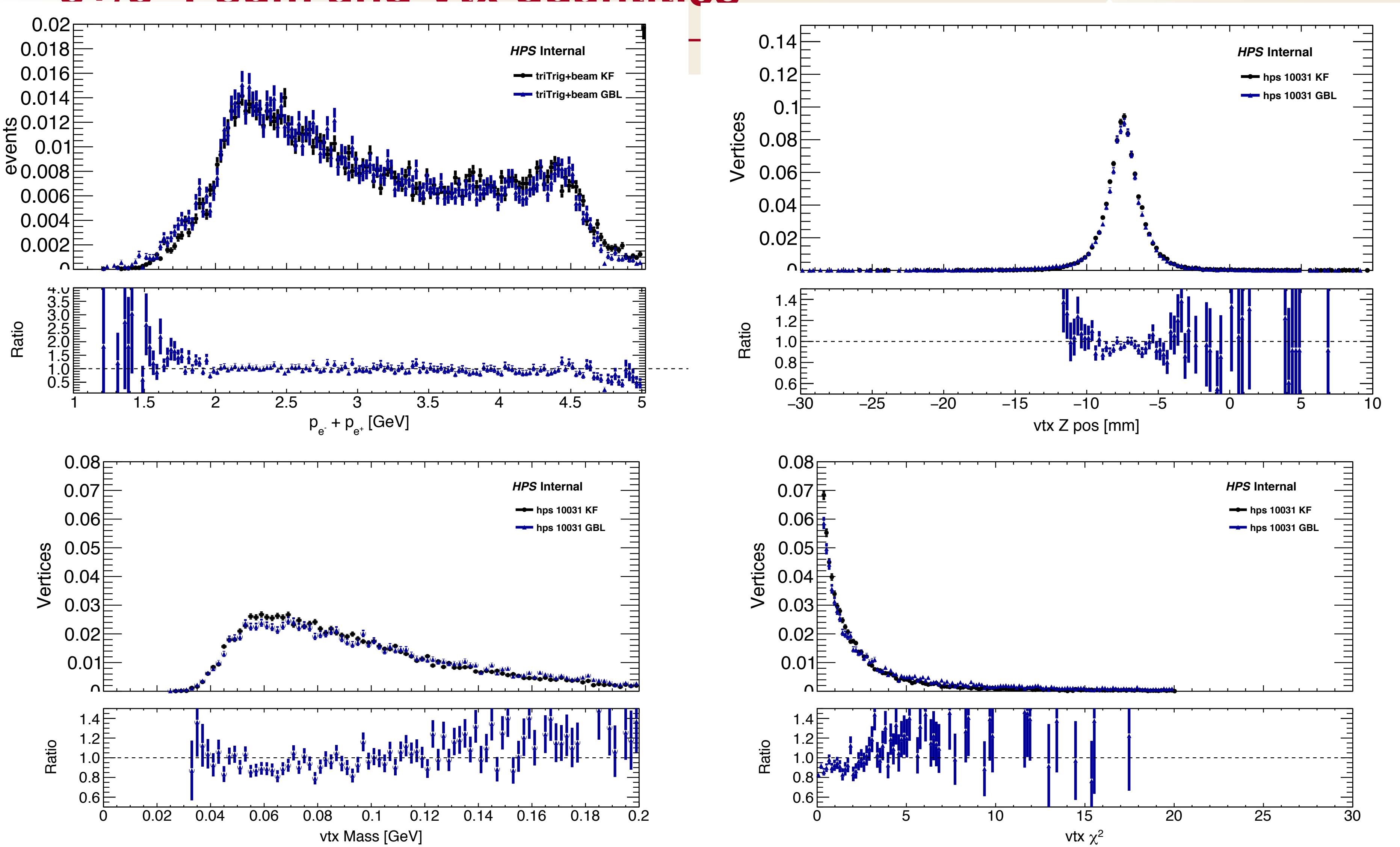




- Quite good agreement between the electron and positron spectrum, after these cleaning cuts
- I placed a cut on the track efficiency plateau for KF tracks. Lower momenta discrepancy maybe due to different turn on curve in this sample? [to be checked]
- Plots normalised to unity



e+/e- Psum and Vtx quantities



- This is just first glance at MC 2019. I didn't have much more time to check other quantities in detail..

Kalman Filter and GBL on Data

- When trying to setup KF on Data I encountered first problems.
- (1) 2019 Data is affected by **Monster Events** which aren't cleaned yet by an appropriate filter
  - Each monster event leads to huge amount of hits in SVT confusing KF Pattern reco
  - SeedTracker has a cut on total number of hits per event at 200, which I now use in KF too
- (2) Several events have O(10^2) seed tracks while in average we find only few tracks per event

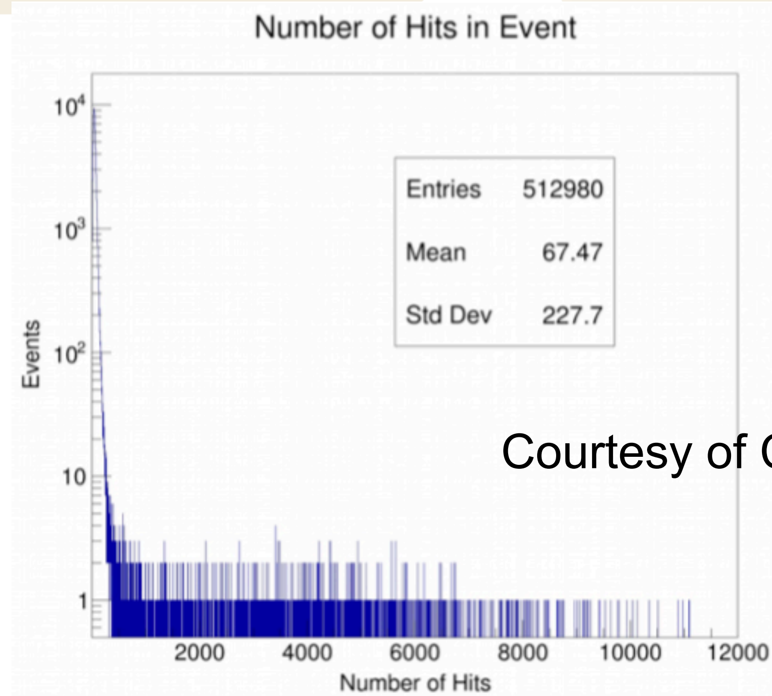
  - Added a check on duplicate seeds

  - Some loss of efficiency (1 track on 5k events)

  - ~10% faster
- **Bottom line:**

  - Data and MC follow a slightly different reconstruction procedure

  - Pattern reco is not tuned on 2019 data



Courtesy of C.Bravo

```
With Seed Duplicate Filter:
2020-04-02 19:15:35 [INFO] org.hps.evio.EvioToLcio run :: maxEvents 5000 was reached
2020-04-02 19:15:35 [INFO] org.lcsim.job.EventMarkerDriver endOfData :: 5000 events processed
in job.
KalmanPatRecDrive.endOfData: total pattern recognition execution time=  55052.7014 ms for 5000
events and 4913 tracks.
```

```
Without Seed Duplicate Filter:
2020-04-02 19:15:46 [INFO] org.hps.evio.EvioToLcio run :: maxEvents 5000 was reached
2020-04-02 19:15:46 [INFO] org.lcsim.job.EventMarkerDriver endOfData :: 5000 events processed
in job.
KalmanPatRecDrive.endOfData: total pattern recognition execution time=  60768.5726 ms for 5000
events and 4914 tracks.
```

Bonus: Checks on 2019 Data Run 10031

SLAC

- Checked Reco Time on Data
- File tested: /nfs/slac/g/hps_data2/data/physrun2019/hps_010031/hps_010031.evio.00054
- **SeedTracker and HelixFitting take ~33%**
- **RawHitFitting takes 32%** of processing time, partly due to monster events rate.
- **KF up to 15%, GBL Refitting 7%**
- **Writing data about 5% of the time**
- **50 seconds for 400 events from evio->LCIO: 0.125s / event**

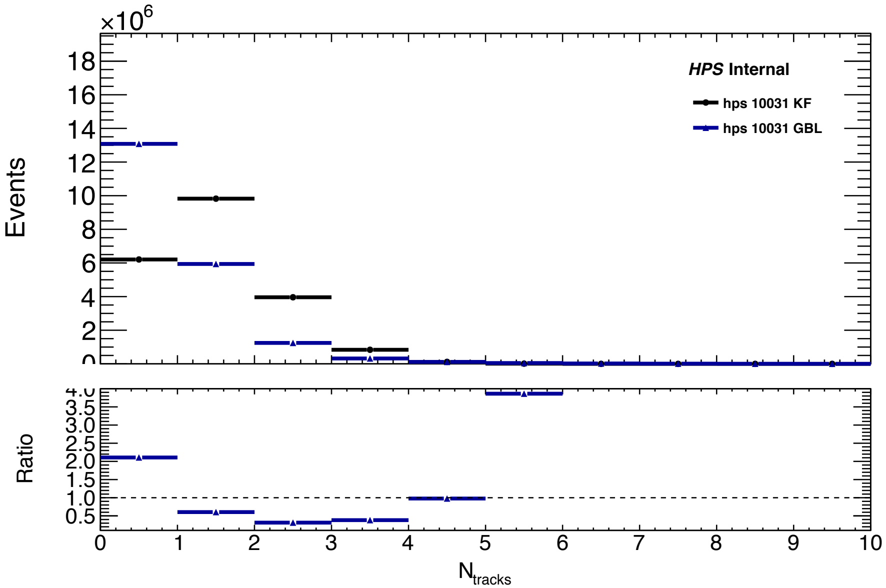
- 99.3% - 101 s org.lcsim.util.Driver.doProcess
  - 33.4% - 34,076 ms org.hps.recon.tracking.TrackerReconDriver.process
  - 31.6% - 32,330 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
  - 14.5% - 14,776 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
  - 6.9% - 7,027 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
  - 5.6% - 5,719 ms org.lcsim.util.loop.LCIODriver.process
  - 3.2% - 3,256 ms org.hps.recon.ecal.EcalRawConverter2Driver.process
  - 1.1% - 1,126 ms org.hps.recon.particle.HpsReconParticleDriver.process
  - 1.1% - 1,075 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
  - 0.8% - 789 ms org.hps.recon.tracking.DataTrackerHitDriver.process
  - 0.6% - 612 ms org.hps.recon.tracking.TrackDataDriver.process
  - 0.3% - 282 ms org.hps.evio.RfFitterDriver.process
  - 0.1% - 142 ms org.hps.recon.ecal.cluster.ReconClusterDriver.process
  - 0.1% - 108 ms org.hps.recon.ecal.HodoRawConverterDriver.process
  - 0.0% - 43,544 µs org.hps.recon.tracking.MergeTrackCollections.process
  - 0.0% - 27,000 µs org.hps.recon.ecal.HodoRunningPedestalDriver.process
  - 0.0% - 21,940 µs org.hps.recon.ecal.EcalTimeCorrectionDriver.process
  - 0.0% - 11,082 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHi
  - 0.0% - 10,886 µs org.lcsim.recon.tracking.digitization.sisim.config.ReadoutClean
  - 0.0% - 10,685 µs org.hps.recon.ecal.EcalRunningPedestalDriver.process
  - 0.0% - 5,311 µs org.hps.recon.ecal.cluster.CopyClusterCollectionDriver.process

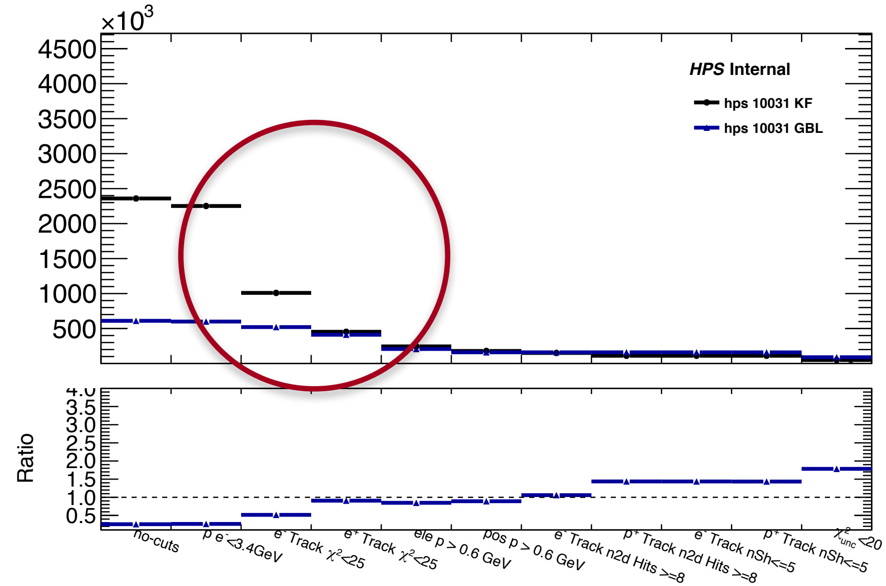
Data processing will be slow with current processing strategy. Something can be

jProfiler

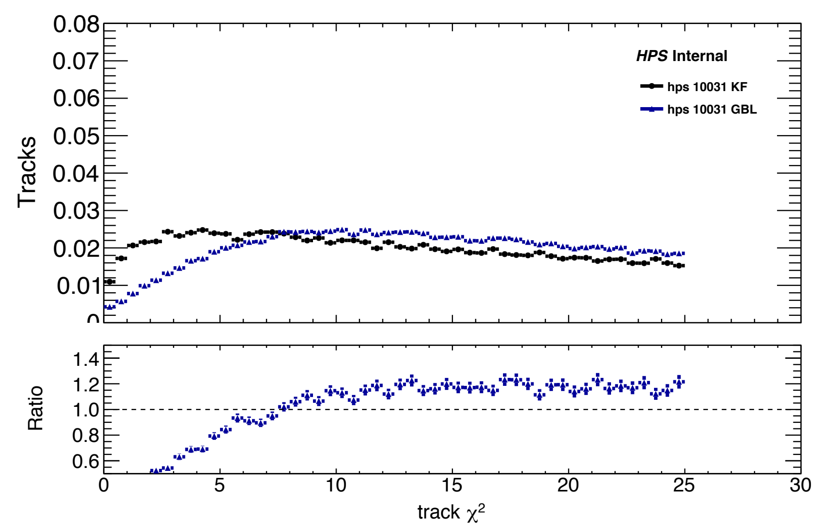
Evaluation version, remotely attached to cent7a, evil to LCIO step

Basic checks on KF/GBL on data

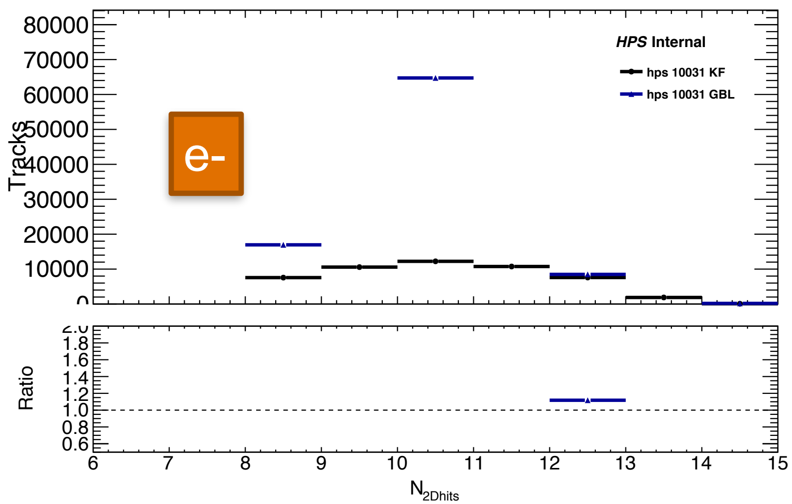


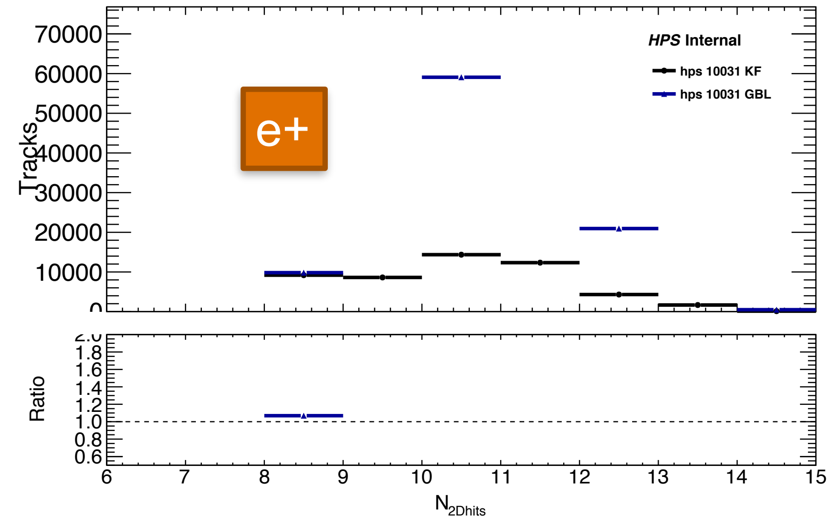


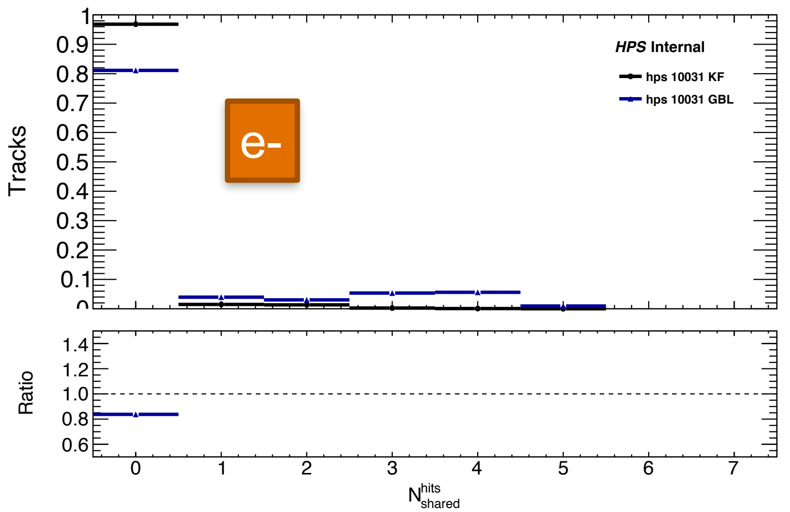
- KF Pattern Reco finds more track wrt GBL in less time
- Chi2 seems to largely reduce the KF tracks.

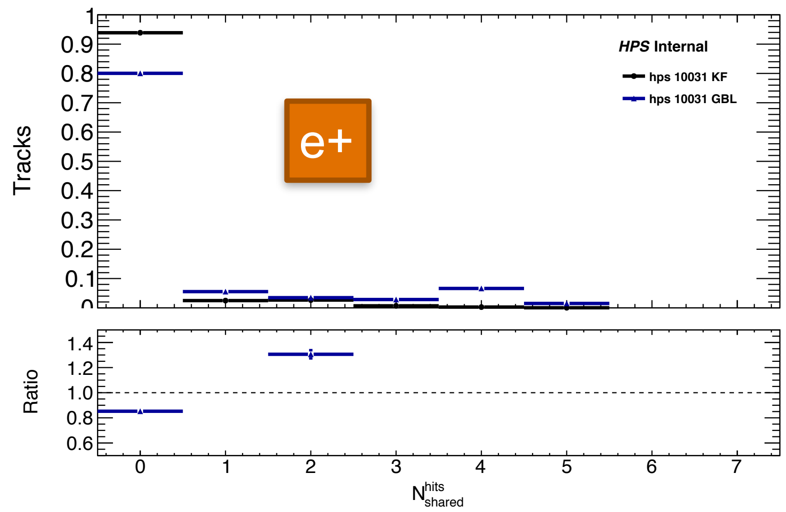


Hit Content in Data 2019 - 10031

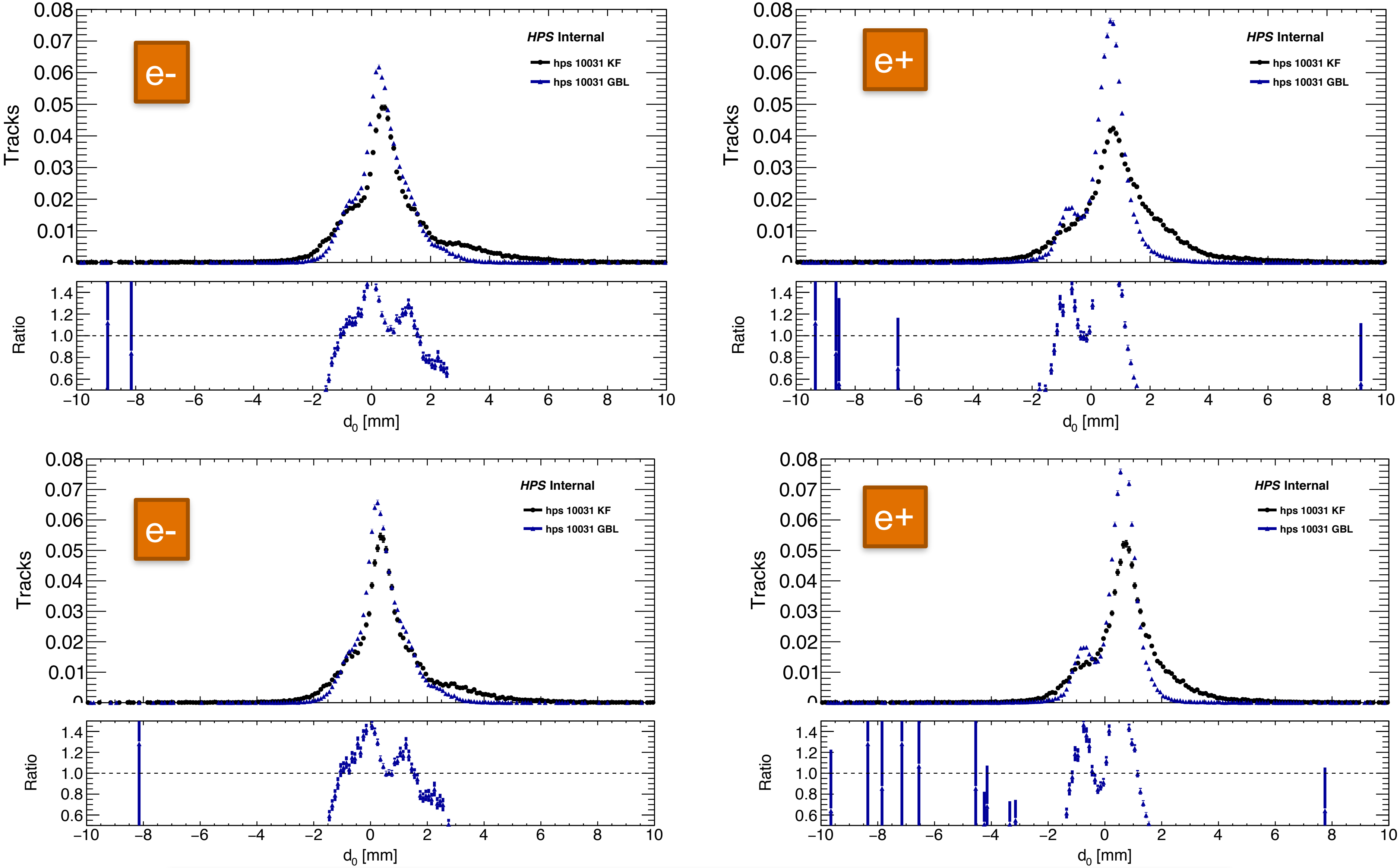






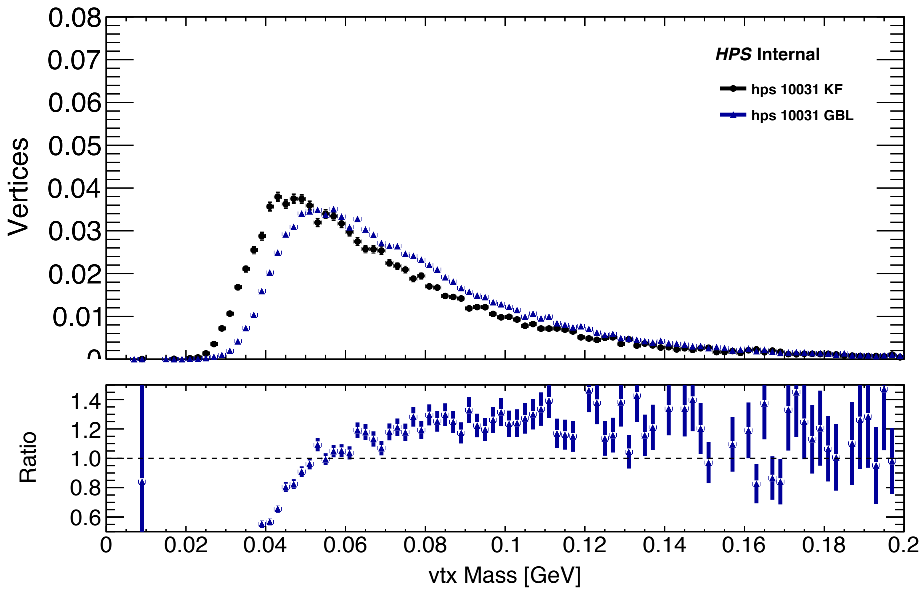


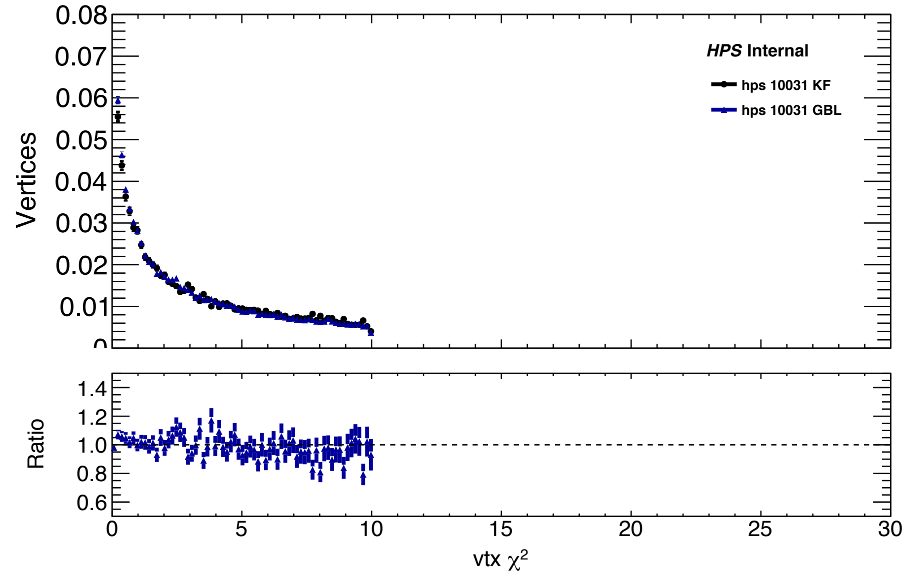
Transverse impact parameter distribution

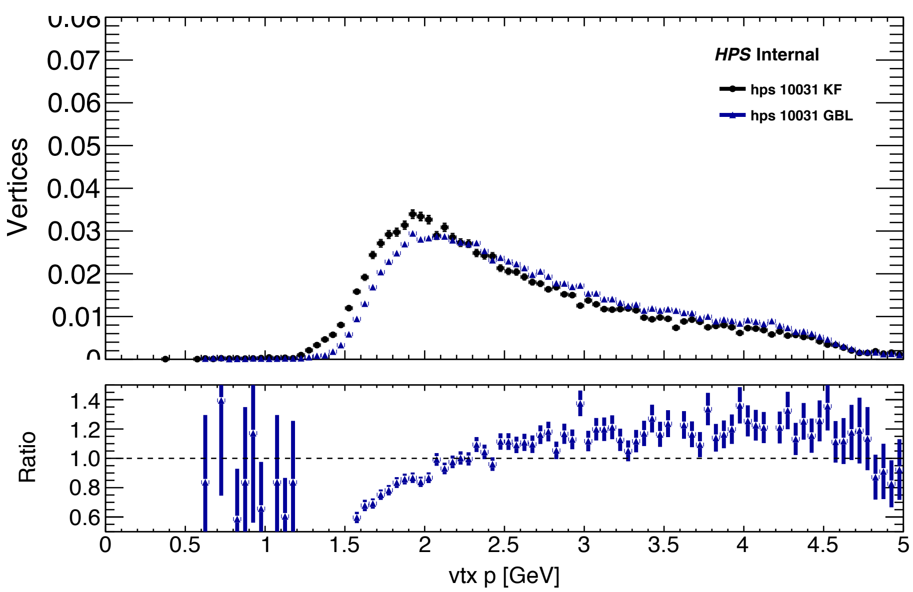


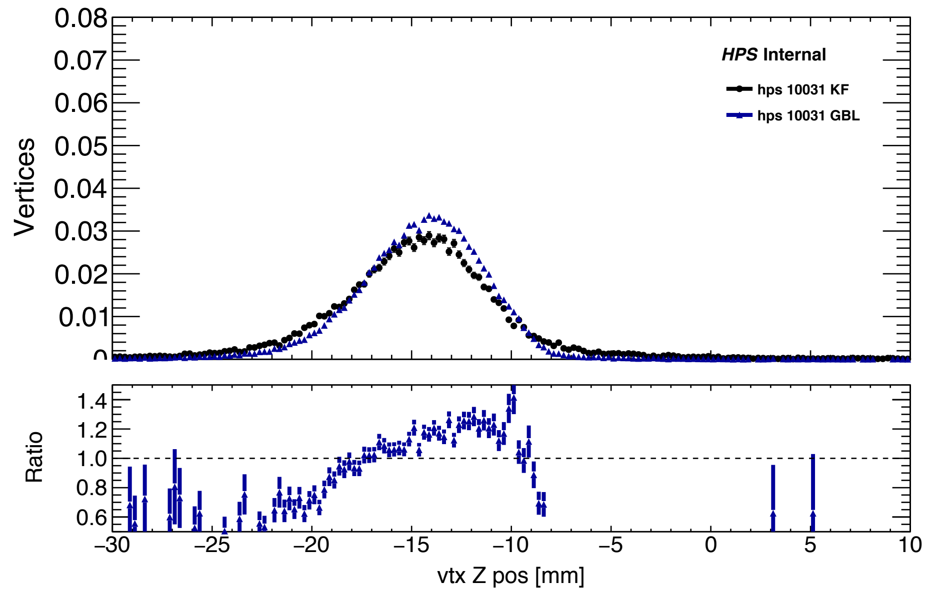
Added Innermost Hit requirement for e+ track in bottom plots [2D for KF, 3D for GBL]

Vertex Properties: Preselection + UncVChi2<10, L0 Hit on e+

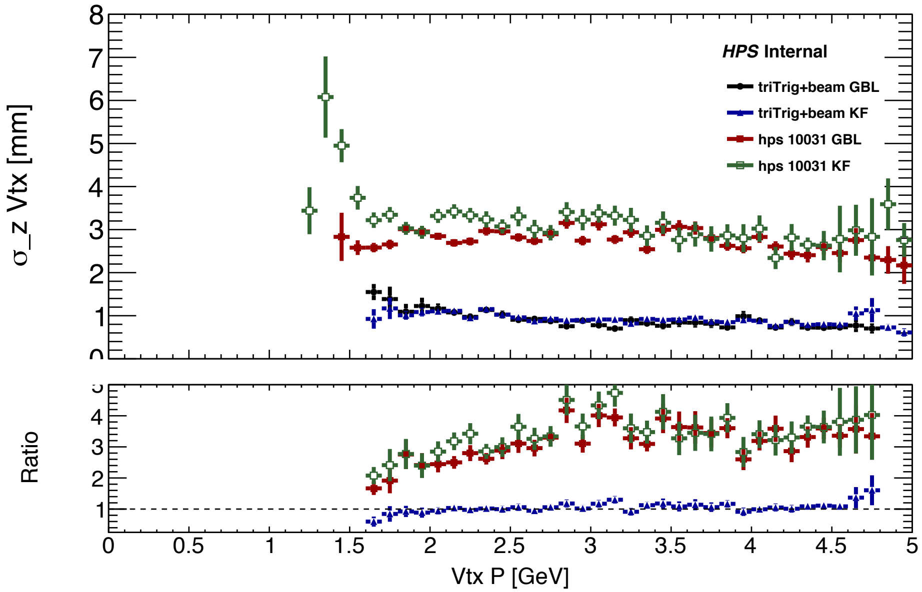


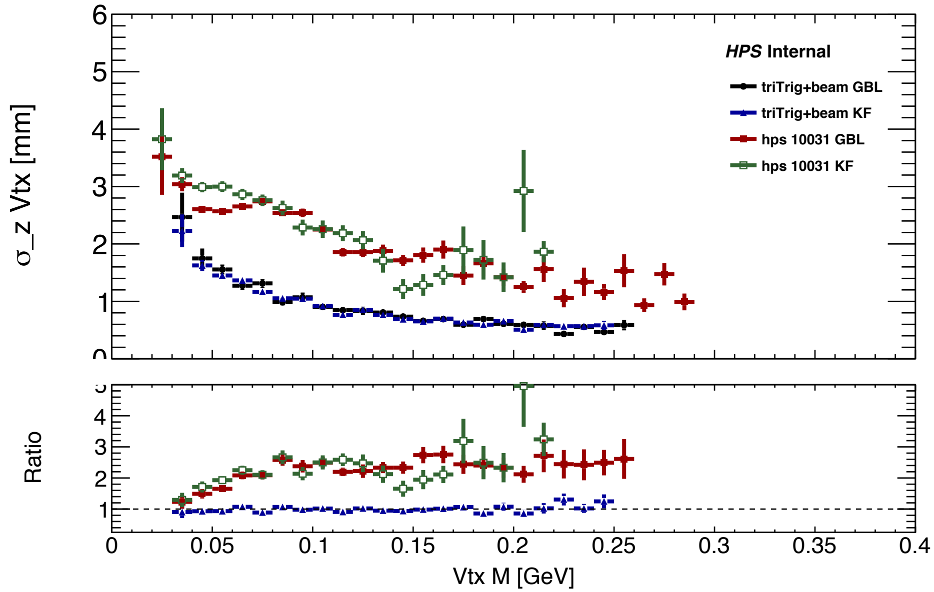


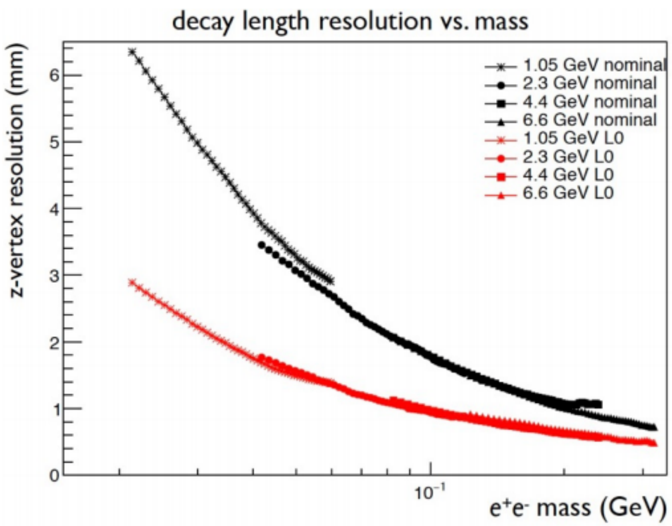




Vertex Resolution - UncVChi2<10, L0 Hit on e+

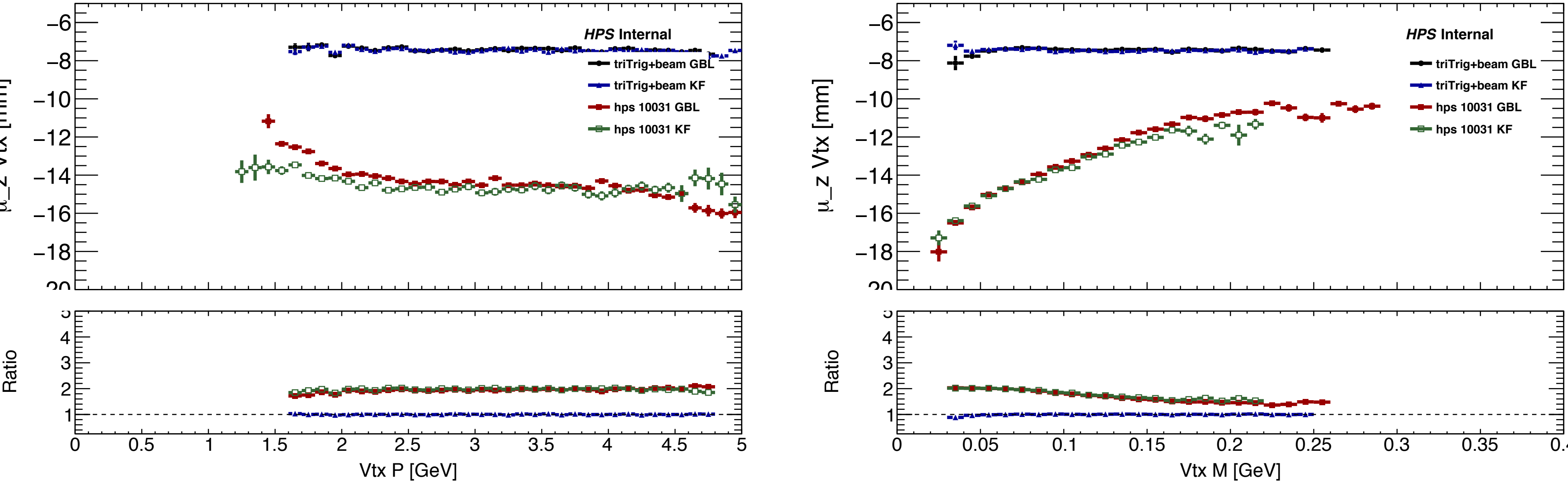






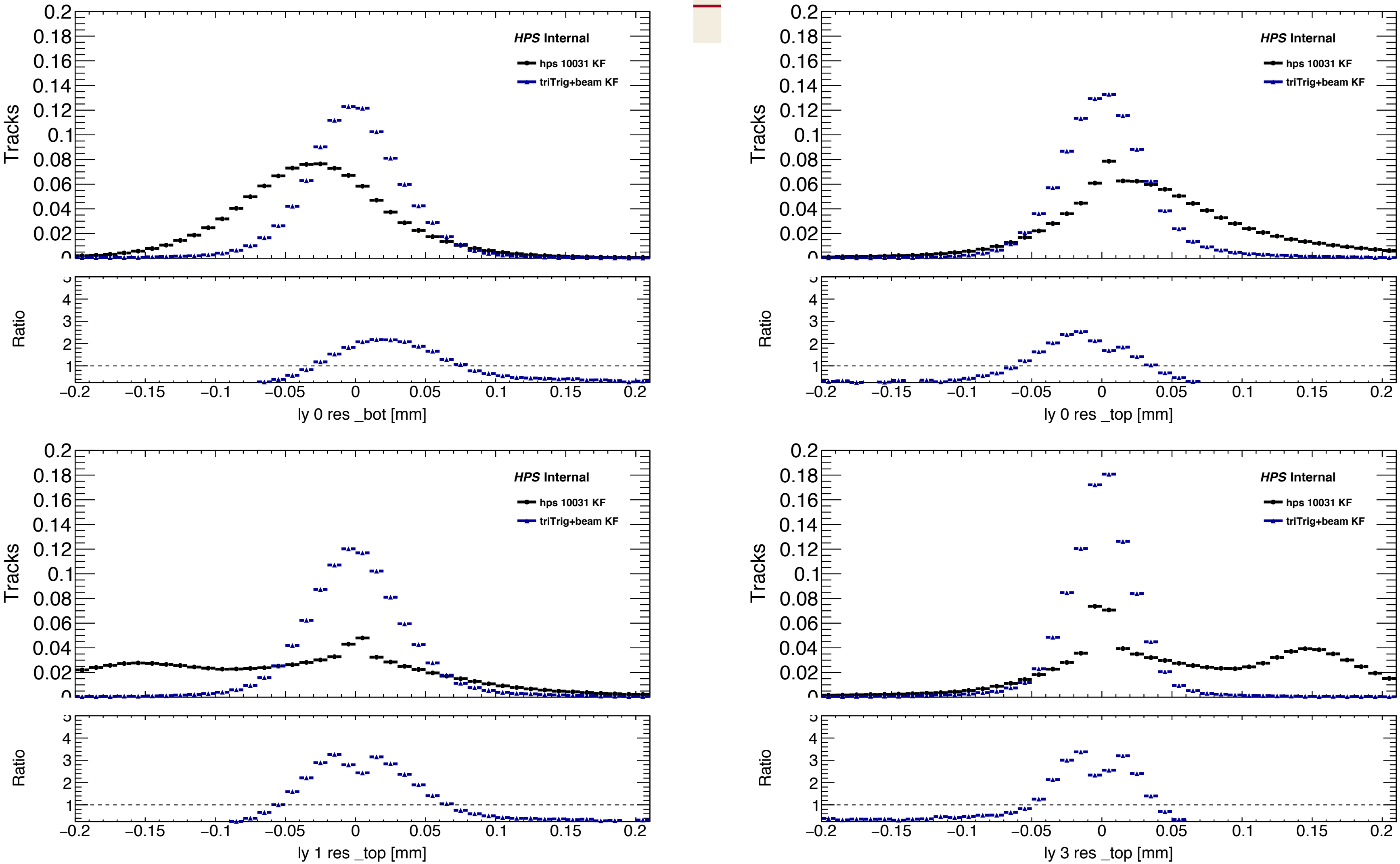
- Obtained by recursive fitting of the gaussian core of the Vtx_Z distribution
- MC reproduces old expected resolution plot (not sure how that was produced back then)
- GBL/Kalman give same results in MC, with better stat for KF tracks
- For Data, seem like KF performs slightly worse, both in term of statistics and extracted resolution. *very* preliminary: pattern reco is not tuned for 2019
- We are factor 3 worse in misaligned detector => top priority

Vertex Mean Drift - UncVChi2<10, L0 Hit on e+



- Steep trend of beamspot position as function of the vertex invariant mass
- KF and GBL tracks show very similar trend in data and MC, with lower stat for KF tracks: again, not tuned for 2019 reco.
- Trends already seen in misaligned 2016 detector => top priority

Hit on Tracks unbiased residuals



Unbiased residuals from Inverse Kalman Filter. To be x-checked with GBL biased/unbiased residuals. Detector V2, no SVT survey

Summary

- Integrated Robert's KF into hps-java reconstruction pipeline
- Seems to perform very well on MC, both with and without beam, however on data 2019 seems like is sub-optimal wrt GBL tracks (from this fast check).
- Work is probably still needed before we can bring this in for analysis, unfortunately

- 2019 Data vertexing performance are ~3x worse (in terms of resolution) wrt expected from MC simulation
- A shift of the mean of Vtx_z position is observed in 2019 data, similar to what was observed in 2016
- With the new siPixel clusters available, alignment is top priority.
- Unbiased residuals from KF (still to be investigated) show large degree of misalignment and bimodal distributions.

BACKUP

Open point for discussion - in random order

- (1) **Proper solution for Monster Events - DATA**
  - We need to fix the SVT Event Filter to remove/skip un-physical events.
  - The current Driver is tuned on 2015 - 2016 studies and need to be fixed for 2019. Current workaround limit of max 200 Clusters/event is arbitrary.
- (2) **MCParticle container in the LCIO is huge** (found about 3k MC Particles per event in the tri-trig + beam)
  - Need to apply cuts before they arrive in final LCIO files
- (3) **Tracking Processing time**
  - Current tracking strategy probably not sustainable in 2019 as takes too much processing time
  - KF pattern reco can be an alternative, once validated and when everyone's happy
- **(4) Raw Hit Fitting Time - DATA (and MC?)**
  - RawSVTHitFittingTime takes 30% of evio->LCIO step in 2019 Data. Can be partially fixed by (1).
  - Alternatively a 2 step process?
    - First we perform a EVIO->FHO [FittedHitsOnly]
    - Hand the FHO for Reconstruction/Alignment/Analysis to people.
      This will cuts 30% of processing time when we'll need to process all the data.
- (5) **BeamSpot determination from Data**
  - BeamSpot info as free parameter are dangerous if BS moves [2016 vex had to recompute it at analysis level]
- (6) **Start an event skimming campaign**
  - A non-negligible amount of events do not even have tracks in them leading to slow processing.
  - Trigger-wise or basic skimming should be done to ensure we don't spend too much time running on useless data. Better earlier than later.
- (7) **Keep track of processing commands**
  - Data we process is often private made with private steering files. We should keep track of what we did in the case of a larger official production.

BACKUP

- **(4) Raw Hit Fitting Time**
  - RawSVTHitFittingTime takes 30% of evio->LCIO step in 2019 Data. Can be partially fixed by (1).
  - A possible compromise while we develop a faster fitting machinery is to reconstruct data in 2 steps:
    - First we perform a EVIO->FHO on the files we want/need [FittedHitsOnly LCIO files] and could start basically today.
    - Use the FHO for Reconstruction/Alignment/Analysis. This will cuts 30% of processing time when we'll need to process all the data.
  - If eventually we get to change fitting we can restart the chain
  - Not directly LCIO as quite slow at the moment and LCIO ntuples content might will change soon.
- (5) **BeamSpot determination from Data**
  - BeamSpot info as free parameter are dangerous if BS moves [2016 vex had to recompute it at analysis level]
  - Propose to do a double processing: x-process and f-process
  - x-process to compute BS information (position/sigma) and store in DB, then f-process for proper correct event-by-event BS/Target constraint.

jProfiler on tri-trig without beam bkg

```
▼ 21.1% - 11,702 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
  ▼ 21.1% - 11,658 ms org.hps.recon.tracking.gbl.MakeGblTracks.refitTrackWithTraj
    ▼ 12.6% - 6,965 ms org.hps.recon.tracking.gbl.MakeGblTracks.doGBLFit
      ▼ 7.1% - 3,926 ms org.hps.recon.tracking.gbl.HpsGblRefitter.fit
        ▶ 3.0% - 1,654 ms org.hps.recon.tracking.gbl.GblTrajectory.<init>
          1.1% - 624 ms java.lang.ClassLoader.loadClass
        ▶ 0.9% - 522 ms org.hps.recon.tracking.gbl.matrix.Matrix.transpose
          0.6% - 310 ms org.hps.recon.tracking.gbl.matrix.Matrix.<init>
        ▶ 0.4% - 197 ms org.hps.recon.tracking.gbl.GlobalDers.<init>
        ▶ 0.3% - 160 ms org.hps.recon.tracking.gbl.GblPoint.<init>
        ▶ 0.2% - 109 ms org.hps.recon.tracking.gbl.matrix.Matrix.inverse
        ▶ 0.2% - 100 ms org.hps.recon.tracking.gbl.GblTrajectory.fit
        ▶ 0.2% - 92,994 µs org.hps.recon.tracking.gbl.matrix.Matrix.copy
        ▶ 0.1% - 65,113 µs org.hps.recon.tracking.gbl.GblPoint.addGlobals
        ▶ 0.1% - 43,746 µs org.hps.recon.tracking.gbl.matrix.Vector.<init>
        ▶ 0.0% - 23,359 µs org.hps.recon.tracking.gbl.matrix.Matrix.times
          0.0% - 11,379 µs java.lang.StringBuilder.append
          0.0% - 5,477 µs org.hps.recon.tracking.gbl.GBLStripClusterData.getTrackPos
      ▼ 5.5% - 3,038 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeStripData
        ▶ 2.5% - 1,402 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeDigiStrip
        ▶ 2.1% - 1,177 ms org.hps.recon.tracking.MultipleScattering.FindHPSScatterPoints
        ▶ 0.8% - 448 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeStripData
          0.0% - 5,553 µs org.hps.recon.tracking.MultipleScattering$ScatterPoints.getScatterPoint
    ▶ 6.7% - 3,684 ms org.hps.recon.tracking.gbl.MakeGblTracks.makeCorrectedTrack(org.hps.recon.
```

```
▼ 11.9% - 6,608 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
  ▼ 11.9% - 6,586 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.prepareTrackCollections
    ▼ 10.4% - 5,778 ms org.hps.recon.tracking.kalman.KalmanInterface.KalmanPatRec
      ▶ 10.0% - 5,550 ms org.hps.recon.tracking.kalman.KalmanPatRecHPS.<init>
```

jProfiler on tri-trig without beam bkg

```
▼ 8.8% - 4,862 ms org.hps.recon.particle.HpsReconParticleDriver.process
  ▼ 8.8% - 4,857 ms org.hps.recon.particle.ReconParticleDriver.process
    ▼ 6.6% - 3,646 ms org.hps.recon.particle.ReconParticleDriver.makeReconstructedParticles
      ▶ 4.1% - 2,249 ms org.hps.recon.ecal.cluster.ClusterUtilities.applyCorrections(org.lcsim.geometry.subdetector.H
```

```
▼ 6.8% - 3,743 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
  ▼ 6.7% - 3,726 ms org.hps.recon.tracking.ShaperPileupFitAlgorithm.fitShape
    ▼ 6.7% - 3,726 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.fitShape
      ▼ 6.7% - 3,726 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.fitShape
        ▼ 6.6% - 3,653 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.doRecursiveFit
          ▶ 5.6% - 3,101 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.minuitFit
          ▶ 1.0% - 535 ms org.hps.recon.tracking.ShaperLinearFitAlgorithm.doRecursiveFit
```

jProfiler on tri-trig with beam bkg

SLAC

jProfiler on tri-trig with beam bkg - rmsTimeCut = 20

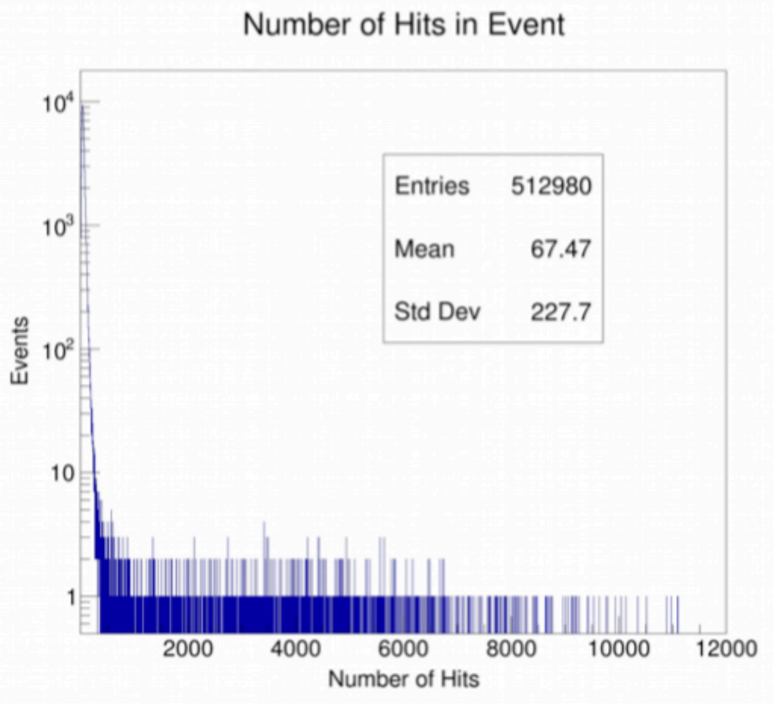
SLAC

```
93.2% - 88,738 ms org.hps.recon.tracking.TrackerReconDriver.process
1.6% - 1,536 ms org.hps.recon.tracking.gbl.GBLRefitterDriver.process
0.8% - 794 ms org.lcsim.util.loop.LCIODriver.process
0.7% - 691 ms org.hps.recon.tracking.kalman.KalmanPatRecDriver.process
0.5% - 518 ms org.hps.recon.tracking.RawTrackerHitFitterDriver.process
0.4% - 373 ms org.hps.recon.ecal.EcalRawConverter2Driver.process
0.3% - 245 ms org.hps.recon.particle.HpsReconParticleDriver.process
0.2% - 149 ms org.hps.recon.tracking.HelicalTrackHitDriver.process
0.1% - 91,609 µs org.hps.recon.tracking.DataTrackerHitDriver.process
0.1% - 48,868 µs org.hps.recon.tracking.TrackDataDriver.process
0.0% - 29,837 µs org.hps.recon.ecal.cluster.ReconClusterDriver.process
0.0% - 27,023 µs org.hps.recon.tracking.MergeTrackCollections.process
0.0% - 16,404 µs org.hps.analysis.MC.TrackToMCParticleRelationsDriver.process
0.0% - 5,556 µs org.lcsim.job.EventMarkerDriver.process
0.0% - 5,490 µs org.lcsim.recon.tracking.digitization.sisim.config.RawTrackerHitSensorSetup.process
0.0% - 5,475 µs org.hps.recon.ecal.cluster.CopyClusterCollectionDriver.process
```

hps-java master issues when running on data

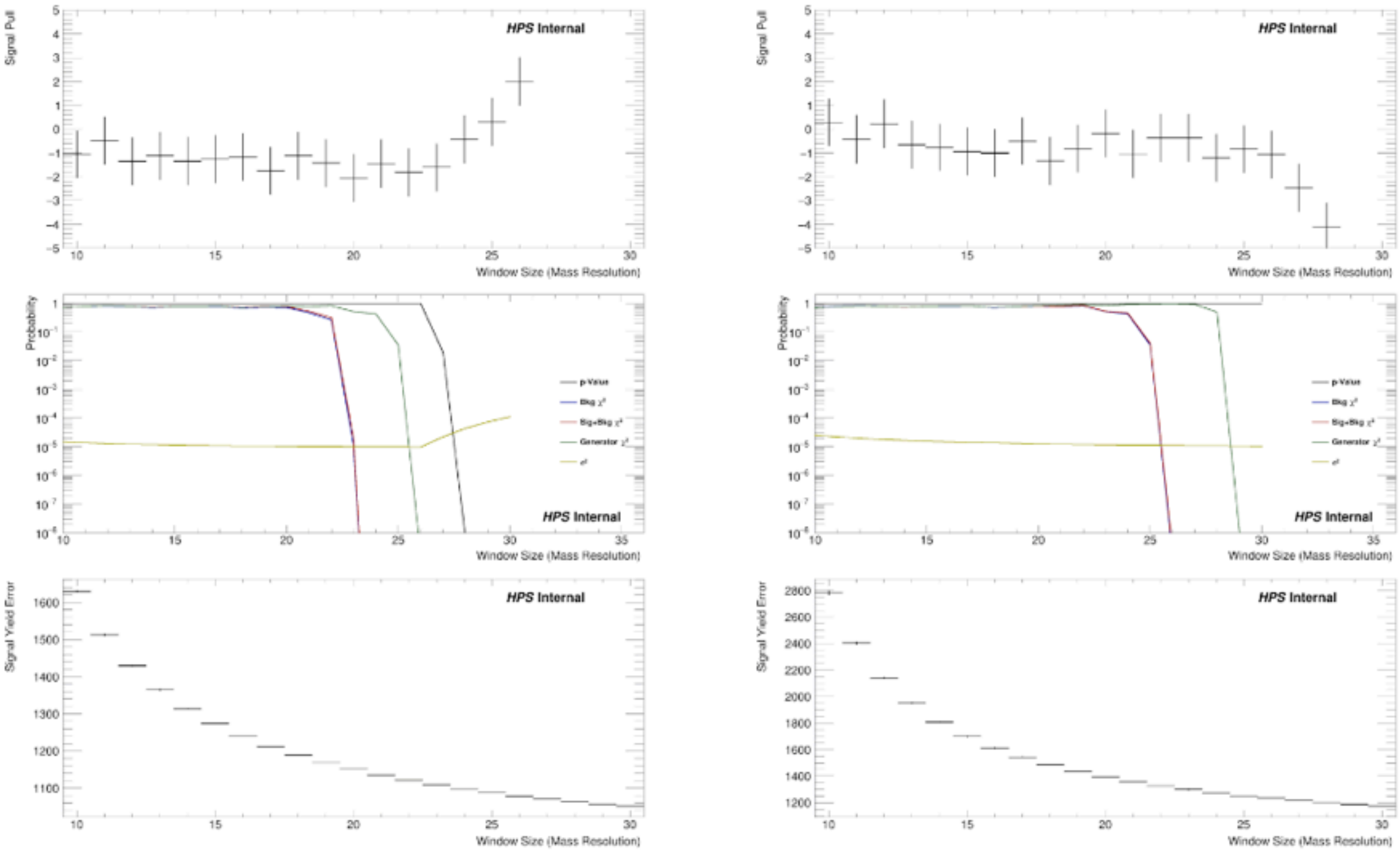
SLAC

- Monster Events:
  - Order of ~% of the events have a huge amount of hits confusing the Track Finding stage
- These event are impossible to process (some lead to more than 10^3-10^4 trackCandidates)
- Current solution
  - Added protection in TrackerHitDriver for SiClusters > 200 [temporary]
  - Added configurable protection on size of **SiClusters** in **KalmanPatDriver** (Same solution of the SeedTracker)



Resonance Search Statistics Support

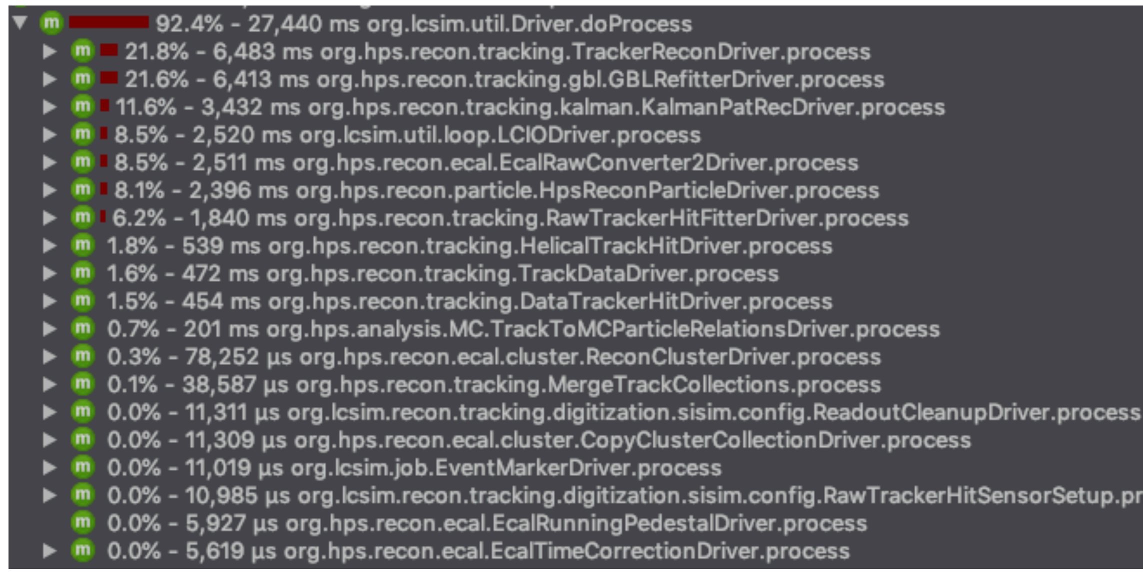
- Fit of mass spectrum and toy model MC with fits fully supported
- Plots for selecting bkg models available



Processing time - Tri-Trig ***without Beam***

SLAC

- A summary breakdown of the CPU time spent in MC processing is shown
- File tested: /nfs/slac/g/hps3/mc/mc_2019/readout/tritrig/singles/4pt5/tritrig_123.slcio
- With the current strategies, tracking takes:
  - ~22% in seeding and global fitting stage
  - ~22% in GBL Refitting stage
- Kalman track finding and fitting takes ~12% of the event time
- Some non-negligible amount of time is spent in the HpsReconParticleDriver (8%) and RawHit Fitting (6%)



Total Tracking time ~40% in tri-trig signal without beam background
See Backup for a more detailed dump

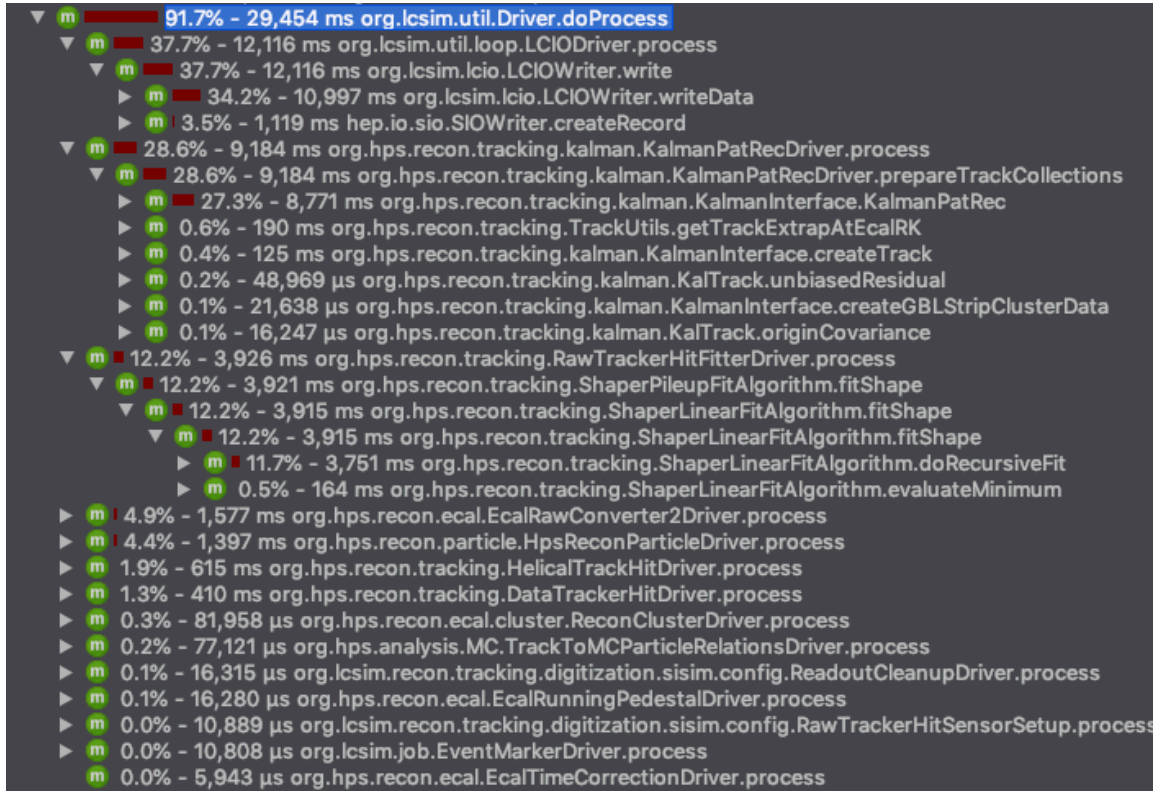
jProfiler

Evaluation version, remotely attached to cent7a, readout to LCIO step

Processing time - Tri-Trig ***with Beam*** KF Only

SLAC

- Tested Kalman only reconstruction
- Kalman track finding and fitting takes ~30% of the event time in this conditions
- Writing LCIO output takes **~40%**
- Something can be recovered from SvtRawHitFitting and HPSReconDrivers
- Only ideal => GBL refitter should run on KF Tracks.
- **Total time: 1m40s for ~860 events on cent7a => 0.11s/ event**



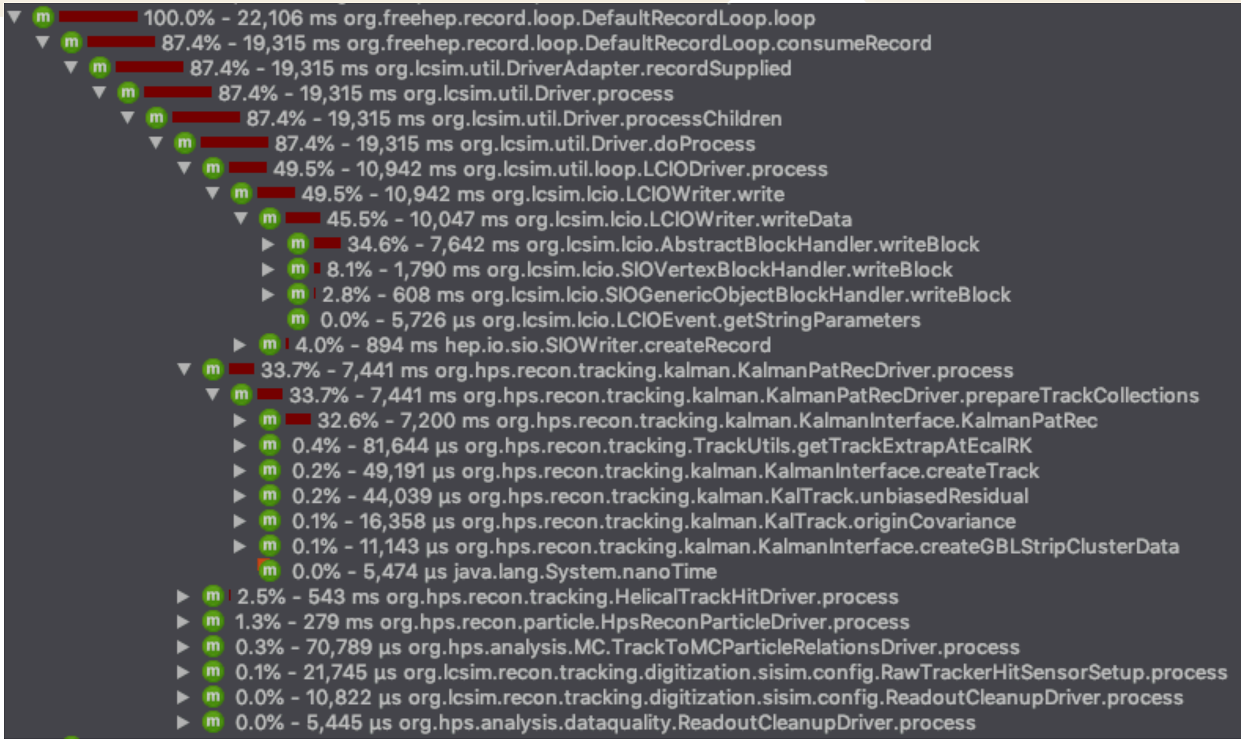
Writing output data is slower than KF tracking, second slowest.
Hit Fitting is a considerable time. Vtxing ~5%

jProfiler

Evaluation version, remotely attached to cent7a, readout to LCIO step

Re-reco from LCIO steering files

- Some time can be saved if running from pre-reconstructed LCIO files, cleaning up proper containers
- I've made a steering file to run on MC from pre-reconstructed LCIO files: iss687_dev => PhysicsRun2019MCRecon_LCIO.lcsim
- Save 12% processing time from RawFitting
- If ran with KalmanOnly: **~0.09s / evt**

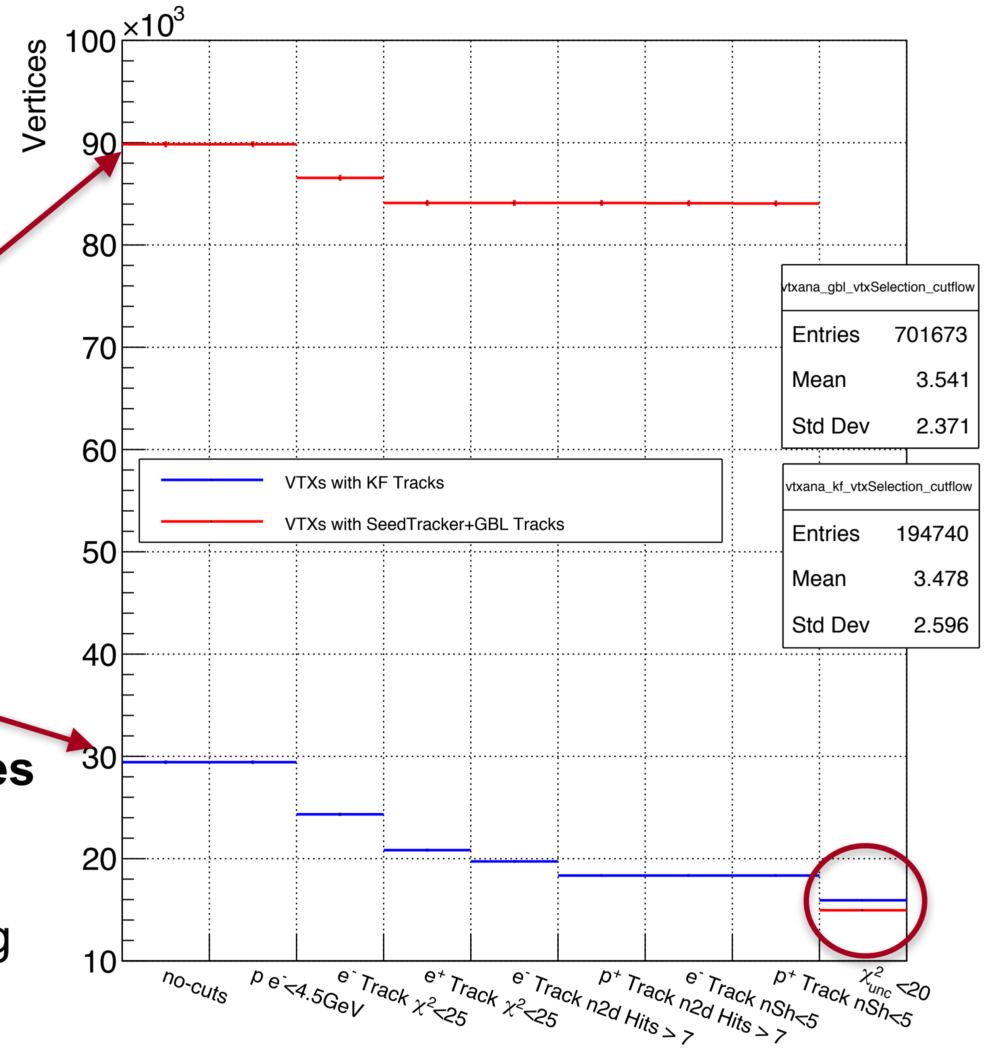


Some fast checks of tri-trig+beam MC 2019

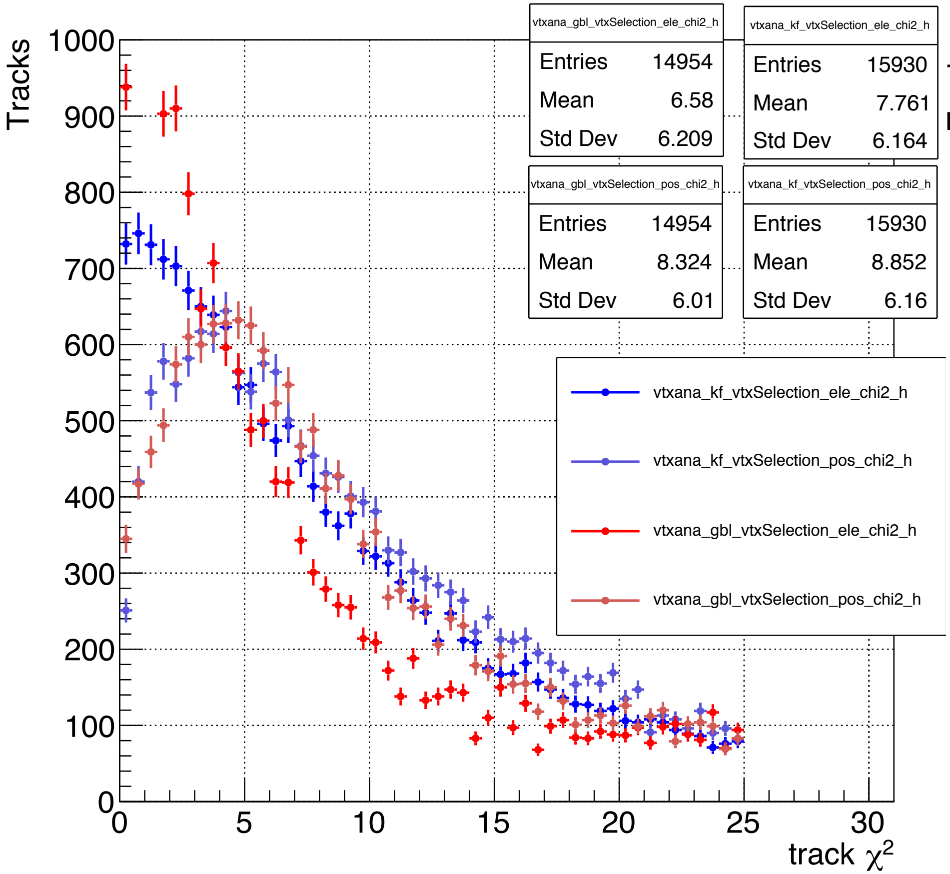
SLAC

- Cuts:
  - ElectronP < 4.5 GeV
  - Ele/Pos Chi2 < 25
  - Ele/Pos n2DHits>=7 [by mistake, should have been >]
  - Ele/Pos nSharedHits<5 [no effect, due to MOUSE]
  - UncVtx Chi2<20

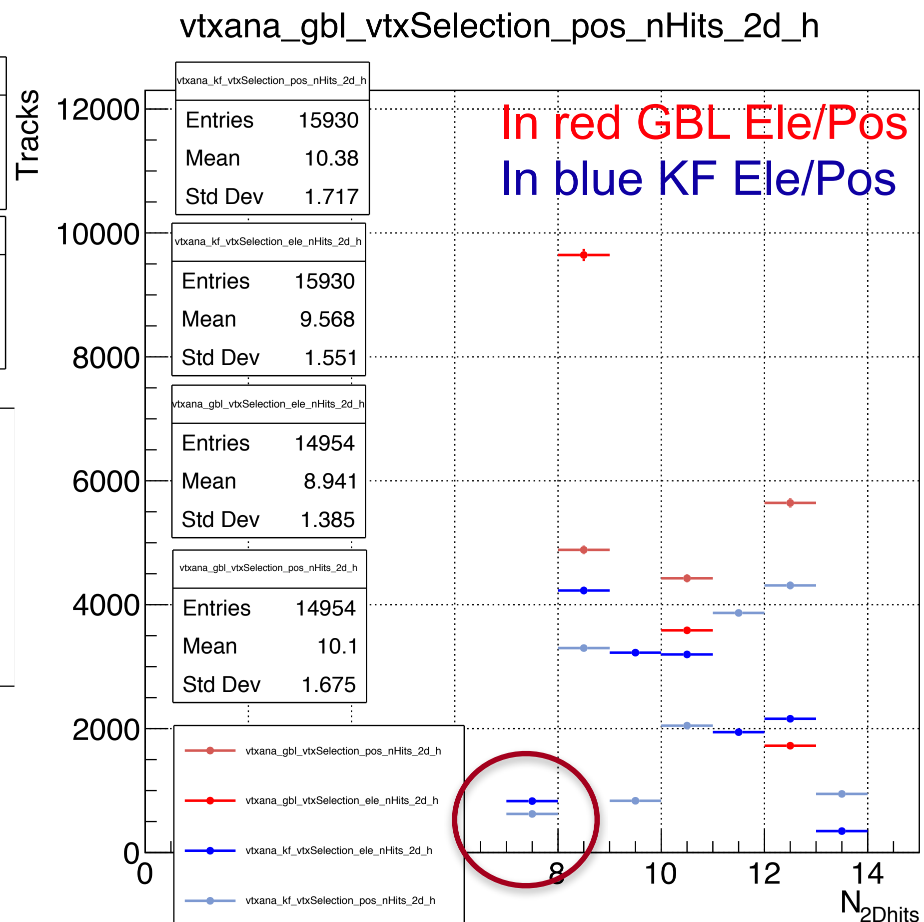
MANY more tracks with SeedTracker, wrt KF to begin with.
However: we know we have lot of lowQuality tracks and duplicates VtxChi2 cleans them all up.
I think this is in line with the long processing time of our standard tracking



Some fast checks of tri-trig+beam MC 2019

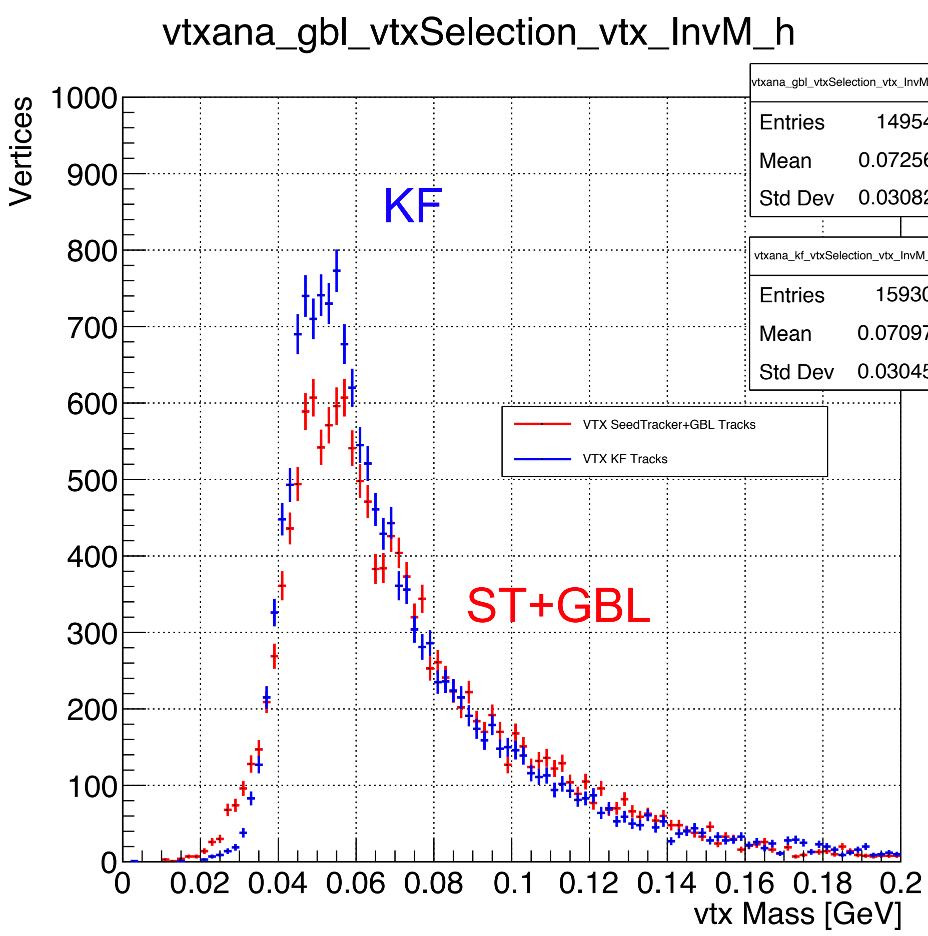


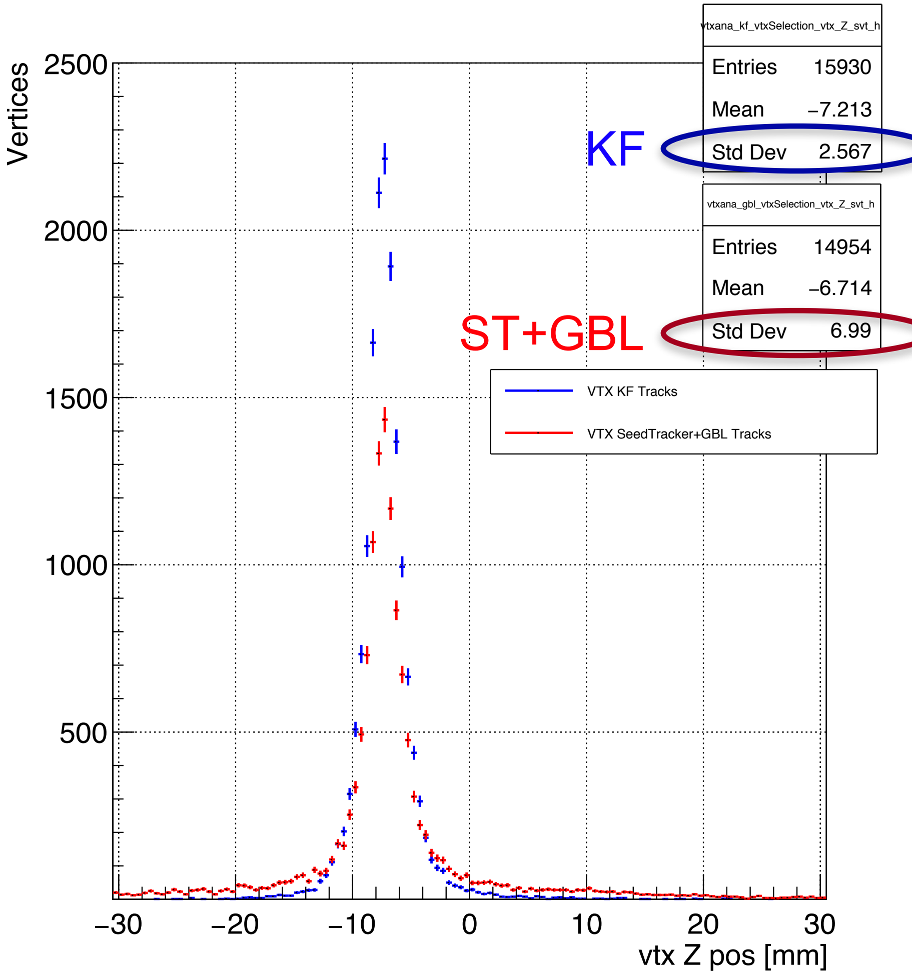
In red GBL Ele/Pos
In blue KF Ele/Pos



Should have removed those

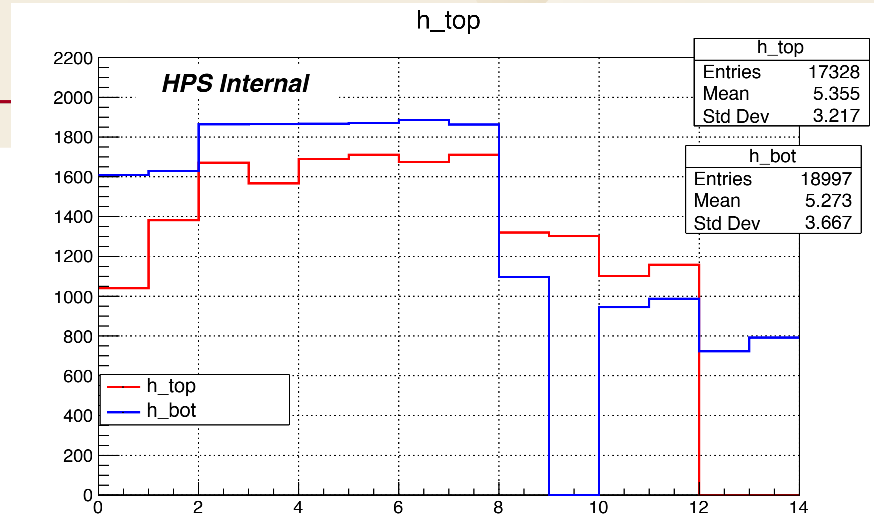
Some fast checks of tri-trig+beam MC 2019





Reconstruction configuration

- The tri-trig readout sample has been generated with:
  - **Top Ly7 (old ly6) off**
  - **Axial Bottom Ly5 (old ly4) off**
- Quite standard job configuration for Hit formation
- Track Finding uses few strategies: only one succeeds for bottom tracks
- To the nominal reco has been added also KF track finding and fitting interfaced with recon drivers
- TrackTruthMatching is provided for offline studies.
  - Tracks are matched to MCParticles which are used to form TruthTracks for performance checks.



HitsOnTrack for KF Tracks

```
<!-- Track finding and fitting using seed tracker. -->

<driver name="TrackReconSeed123Conf4Extd56"/>
<driver name="TrackReconSeed123Conf5Extd46"/>
    <driver name="TrackReconSeed567Conf4Extd123"/>
<driver name="TrackReconSeed456Conf3Extd127"/>
<driver name="TrackReconSeed356Conf7Extd124"/>
<driver name="TrackReconSeed235Conf6Extd147"/>
```

only one that succeeds for bottom

```
<driver name="MergeTrackCollections"/>
<driver name="GBLRefitterDriver" />
<driver name="TrackDataDriver" />
<driver name="KalmanPatRecDriver"/>
<driver name="TrackTruthMatching_KF" />
<driver name="TrackTruthMatching_GBL" />
<driver name="ReconParticleDriver" />
<driver name="ReconParticleDriver_Kalman" />
<driver name="LCIOWriter"/>
<driver name="CleanupDriver"/>
```

For KF tracks, Vtxing finalStateParticles

Nominal Helix+GBL

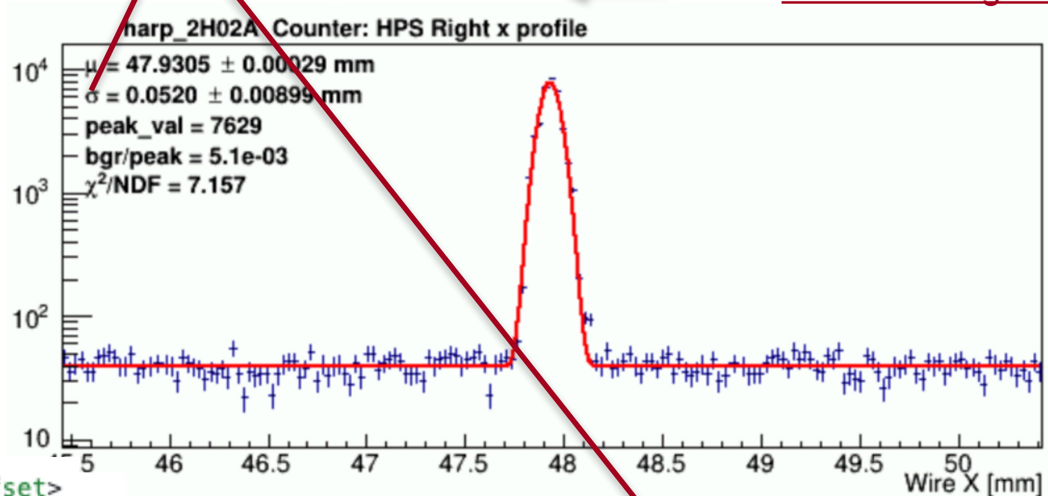
For truth links in LCIO outfile

Reconstruction configuration - ReconParticleDriver

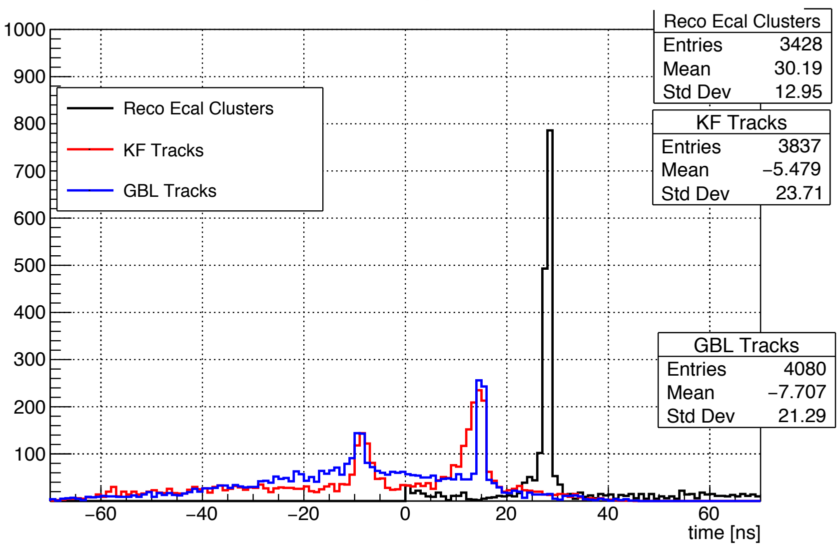
```
<beamPositionX> 0  </beamPositionX>
<beamSigmaX> 0.05 </beamSigmaX>
<beamPositionY> 0 </beamPositionY>
<beamSigmaY> 0.02 </beamSigmaY>
<beamPositionZ> -7.5 </beamPositionZ>
```

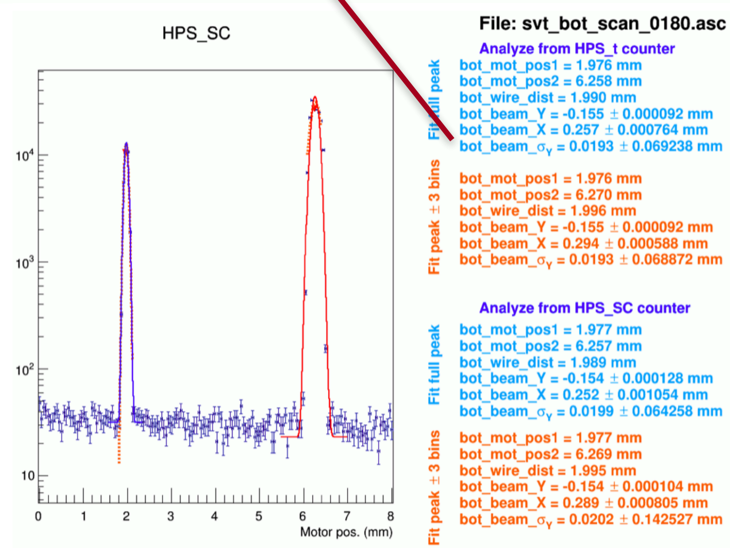
From MC sim configuration

- **Vertices** are formed with both Helix+GBL and KF Tracks
  - *Vertices formed with KF Tracks have "_KF" in CollectionName*
- **Vertices are formed without requiring cluster/track matching**
  - *Still working to check track-cluster matching in 2019*
- Nominal settings for BS position (0,0,-7.5)
  Size was taken from SVT wire scan to resemble data (simulation was done with sigma_x(y) = 0 mm)
- TrackClusterTimeOffset from checking ClusterTime distribution
  - Tracks time distribution needs to be cross checked as double peak wasn't expected




```
<trackClusterTimeOffset>28</trackClusterTimeOffset>
```





Ele / Pos momenta: “Tight: UnCVChi2<10, L0 hit on e+”